Procesamiento del Lenguaje Natural

Presentación de la asignatura

LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

Agenda

- Presentaciones
- Objetivos de la asignatura
- Contenidos y estructura
- Actividades puntuables
- Evaluación

Presentaciones

Universidad Internacional de La Rioja

Acerca de…

Alberto Barbado González

- Doctor en Inteligencia Artificial
- Breve trayectoria: Senior Data Scientist (sector bancario), Profesor del Master de IA en la UNIR
- Perfil de investigación: Explainable AI, Text Mining, Anomaly Detection
- Me encargo de
  - Clases teóricas del temario
  - Clase de tutoría sobre el examen
  - Foro de dudas

Acerca de…

Javier Hernádez Afonso

- Profesor Corrector
- Se encarga de
  - Sesiones de Refuerzo
  - Correcciones (Actividades y Examen)

Acerca de…

Vosotros

- ¿Desde qué ciudad te conectas?
- ¿Qué esperas de esta asignatura?
- Una palabra que relaciones con PLN

Objetivos de la asignatura

Objetivos de aprendizaje

- **Identificar** diferentes ámbitos de aplicación del procesamiento de lenguaje natural
- **Distinguir** los bloques funcionales básicos de una aplicación que utiliza procesamiento de lenguaje natural
- **Construir** aplicaciones básicas que utilicen técnicas de procesamiento de lenguaje natural
- **Seleccionar** el modelo de PLN más adecuado a las necesidades de un proyecto.

Contenidos y estructura

Sobre la asignatura: contenido

- T1: Introducción
- T2: El Texto como Dato
- T3: Etiquetado Morfosintáctico *(POS Tagging)*
- T4: Análisis Sintáctico
- T5: Análisis Semántico
- T6: Semántica Léxica
- T7: Modelado Estadístico del Lenguaje
- T8: Modelado Neuronal del Lenguaje
- T9: Aplicaciones del PLN
- T10: Agentes conversacionales

Sobre la asignatura: planificación

Actividades puntuables

Actividades

- Actividad individual: Caracterización de textos
- Laboratorio: Word Embeddings y Transformers para clasificación de texto
- Actividad grupal: Aplicaciones prácticas de PLN en investigación y en el mercado final

Test de autoevaluación

- Hay un test en cada tema

- El objetivo es que sirvan de repaso, pero también de incentivo para el estudio

Asistencia a clases

- Criterio: asistes si estás más de 30 minutos en clase
- Puntúan un máximo de dos sesiones

Foro de consultas

- NO puntúa
- Se agradece la participación
- Tiempo de respuesta aproximado: 24 horas

Evaluación

Sobre la asignatura: evaluación

- Evaluación continua (40%)
  - Actividades
    - Trabajo Individual (5)
    - Laboratorio (5)
    - Actividad grupal (3,5)

    Las actividades no aprobadas, no suman

    - Asistencia a presenciales (0.25 +0.25)
    - Test de autoevaluación (0.1 x 10)

- Examen (60%)

  Es obligatorio aprobar el examen

Sobre la asignatura: evaluación

Solo suman aquellas actividades aprobadas (>=5).
Las actividades valoradas <5 sumarán 0 puntos en la evaluación.

Solo suman aquellas actividades aprobadas en ordinaria.
No se entregan actividades en extraordinaria.

unir

LA UNIVERSIDAD
EN INTERNET

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 1 - Introducción

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Encuesta previa

- ¿Recuerdas las clases de lengua del colegio e instituto?
- Nombra una aplicación que conozcas que usa PLN
- Nombra una tarea que conozcas que se realiza con PLN

En el día de hoy

- Definición de Procesamiento de Lenguaje Natural
- Aplicaciones de PLN
- Lingüística y PLN
- Historia del Procesamiento de Lenguaje Natural

Definición de Procesamiento de Lenguaje Natural

Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do. (IBM).

El procesamiento de lenguaje natural es un campo de las ciencias de la computación, de la inteligencia artificial y de la lingüística que estudia las interacciones entre las computadoras y el lenguaje humano. (Wikipedia)

Campo de la Inteligencia Artificial que tiene como objetivo principal hacer que las máquinas sean capaces de realizar tareas que involucren el lenguaje humano (Apuntes de la asignatura)

Otros Nombres

- Procesamiento de Lenguaje Natural (PLN)
- Natural Language Processing (NLP)
- Procesamiento del Lenguaje y el Habla
- Tecnología del Lenguaje
- Lingüística computacional
- Reconocimiento y Síntesis del Habla

Aplicaciones de PLN

- Information Retrieval
- Question Answering
- Resumen Automático
- Traducción Automática
- Generación de Textos
- Language Modeling
- Named Entity Recognition
- Análisis Sintáctico
- Análisis Morfosintáctico (POS Tag)
- Word Sense Disambiguation
- Speech Recognition
- Speech Generation
- Sistemas de Diálogo

Aplicaciones de PLN

Natural Language Processing

- POS tagging
- Semantic Role labeling
- Named Entity Recognition

Natural Language Understanding

- Question Answering
- Text similarity

Natural Language Generation

- Sumarization
- Translation

Sentiment Analysis

Conversational Agents

Lingüística y PLN

Fonética y fonología	Morfología	Sintaxis
Semántica	Pragmática	Discurso

Figura 7. Conocimientos necesarios para tareas complejas de PLN.

Lingüística y PLN

- **Fonética**: “conocimiento de cómo se pronuncian las palabras a partir de la secuencia de sonidos”
- **Morfología**: “forma en que las palabras se descomponen en partes que tienen un significado”
- **Sintaxis**: “conocimiento necesario para ordenar y agrupar palabras”
- **Semántica**: “conocimiento sobre el significado de cada una de las palabras, su composición, o el significado de varias palabras conjuntas. Se incluye el conocimiento de la relación de las palabras con la estructura sintáctica (e.j., sintagmas)”
- **Pragmática:** “conocimiento de la relación del significado con los objetivos y las intenciones (e.j., es una pregunta, una afirmación…)”
- **Discurso**: “conocimiento sobre el discurso previo para resolver las referencias cruzadas”

Historia del Procesamiento de Lenguaje Natural

1940 1950	1957 1970	1970 1983	1983 1993	1993 1999	2000 2016	2016…
Automatas Modelos prob.	**Simbólico** Estocástico	**Estocástico** Lógica Comprensión Discurso	Estados finitos **Empirismo**	**Empirismo**	**Aprendizaje automático** Redes neuronales	**Deep Learning**

Historia del Procesamiento de Lenguaje Natural

Métodos formales

- Teoría de lenguajes formales:
  - Un lenguaje **L** es un conjunto, finito o infinito de secuencias **s**
  - Un vocabulario **V** es el conjunto de símbolos disponibles en el lenguaje
  - Una secuencia **s** es una concatenación de símbolos de **V**
  - Una gramática **G** es una definición formal de **L**: establece qué combinaciones de **V** forman secuencias **s** válidas en **L** y cuáles no

Métodos formales

- Teoría de autómatas:
  - Definición de lenguajes formales equivalente
  - Mediante máquinas abstractas
  - Formadas por un conjunto de estados **S** y una función de transición **T** que define transiciones de un estado a otro en función de los símbolos de una cadena de entrada

Métodos formales

Métodos formales

– ¿Puede el lenguaje natural formalizarse como un lenguaje formal **L**, mediante una gramática **G**, o como un autómata?

Métodos probabilísticos

- N-gramas
- Cálculo de probabilidades
- Cadenas de Markov (Hidden Markov Models o HMM)

Métodos probabilísticos

- N-gramas

  “por favor, por favor, no olvidemos que por este camino ya hemos pasado antes”

(por favor) 2

(favor ,) 2

(, por) 1

…

(por este) 1

(este camino) 1

…

Métodos probabilísticos

- Cálculo de probabilidades de N-gramas

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

Métodos probabilísticos

- #ocurrencias **por favor** en el corpus: 2
- #ocurrencias **por este** en el corpus: 1
- #ocurrencias **por** en el corpus: 3

$$P(favor|por) = \frac{C(porfavor)}{C(por)} = 0.66$$

$$P(este|por) = \frac{C(poreste)}{C(por)} = 0.33$$

Métodos probabilísticos

- Autómatas con probabilidades de transición

Machine Learning

- Machine Learning / Deep Learning
- Aprendizaje supervisado
- Proliferación de datos. Internet, redes sociales
- Soluciones de hardware: cloud, GPU, TPU
- Permiten modelar patrones demasiado complejos para formalizarlos explícitamente

Historia del PLN: aprendizaje automático

- Deep Learning
- Redes recurrentes: RNN, LSTM
- También convolucionales: CNN
- Transformadores: BERT
- Transfer Learning
- GAN aplicado a PLN

En resumen

- Aplicaciones de PLN
  - NLP (etiquetado),
  - NLU (question answering)
  - NLG (text generation)
- Lingüística y PLN
  - Morfología, sintaxis, semántica, pragmática, discurso
  - Fonética
- Historia del Procesamiento de Lenguaje Natural
  - Modelos formales
  - Modelos probabilísticos
  - Modelos basados en Machine Learning

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 2 – El texto como dato

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Encuesta previa

- ¿Sabes qué es un Notebook de Python?
- ¿Tienes instalado Jupyter?
- ¿Sabes usar Google Colab?
- ¿Sabes qué es un corpus lingüístico?
- ¿Sabes la diferencia entre *lema* y *lexema*?

Índice

- Principios conceptuales
- Morfología
- El pipeline de normalización
- Recursos
- Herramientas y librerías

Principios conceptuales

Principios conceptuales

- Idea principal
  - La estructura del lenguaje natural es demasiado compleja
- Solución
  - Simplificar los textos de forma sistemática

Principios conceptuales

- Simplificamos en base a tres principios:
  - Principio de composición
  - Aspecto secuencial
  - Perspectiva distribucional

Principio de composición

A nivel de oración:

El significado de una oración se construye como composición del significado de las palabras que contiene

"Me gustan los libros de fantasía" ⇨

- Me
- gustan
- los
- libros
- de
- fantasía

Principio de composición

A nivel de oración:

El significado de una oración se construye como composición del significado de las palabras que contiene

A nivel de palabra:

La información de una palabra se construye como composición de la información que aportan sus morfemas

“perros” -> “perro” + “s”

Aspecto secuencial

El orden en el que aparecen las palabras importa en la construcción del significado de una oración

“Un hombre pobre”

Vs

“Un pobre hombre”

“Al campo no he ido, he ido a la playa”

Vs

“A la playa no he ido, he ido al campo”

Perspectiva distribucional

El significado de una palabra en un texto depende de las palabras que la rodean

- Resulta útil en la solución de ambigüedades
  - … fue al **banco** a contratar una hipoteca.
  - … en el mar pude ver un **banco** de peces rojos.
  - … ellos se sentaron en un **banco** de madera.

Morfología

Principio composicional aplicado a palabras

Morfología de las palabras

- Morfemas :
  - **Lexema / morfema léxico**: raíz de la palabra con significado léxico
  - **Morfema gramatical**: significado gramatical - género, número, tiempo, etc.
    - **Morfema cero**: marca gramatical sin marca fonética (ej. mujer)
  - **Lema**: unidad autónoma con el significado léxico raíz de la palabra

Morfología de las palabras

- Ejemplos

Palabra	Lexema	Morfema gramatical	Morfema gramatical	Lema
Cantábais	Cant	ába	is	Cantar
Casita	Cas	ita	Morfema cero (SG)	Casa
Perros	Perr	o	s	Perro
Vino	Vino	Morfema cero (M)	Morfema cero (SG)	Vino

Morfología de las palabras

```
[ ] !python -m spacy download es_core_news_md
```

```
[1] import spacy
    import es_core_news_md
```

```
[2] nlp = es_core_news_md.load()
    doc = nlp("Una historia de un perro y unos gatos")
```

```
[3] for token in doc:
      print("\"", token, "\"", token.morph)
```

```
" Una " Definite=Ind|Gender=Fem|Number=Sing|PronType=Art
" historia " Gender=Fem|Number=Sing
" de "
" un " Definite=Ind|Gender=Masc|Number=Sing|PronType=Art
" perro " Gender=Masc|Number=Sing
" y "
" unos " Definite=Ind|Gender=Masc|Number=Plur|PronType=Art
" gatos " Gender=Masc|Number=Plur
```

El pipeline de normalización

Pipeline

- Tokenización
- Borrado de StopWords
- NER
- Mayúsculas y minúsculas
- Lematización

Pipeline (detalle)

- Tokenización
  - “En un lugar de La Mancha, de cuyo nombre no quiero acordarme”
  - [“En”, “un”, “lugar”, “de”, “La”, “Mancha”, “,”, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]

```
In [4]: import spacy
        import es_core_news_sm
```

```
In [5]: nlp = es_core_news_sm.load()
        doc = nlp("En un lugar de La Mancha, de cuyo nombre no quiero acordarme")
```

Pipeline (detalle)

- Borrado de stopwords
  - [“En”, “un”, “lugar”, “de”, “La”, “Mancha”, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]
  - [“lugar”, “Mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]

```
for token in doc:
    if not(token.is_stop):
        print(token)
```

```
lugar
Mancha
,
cuyo
nombre
quiero
acordarme
```

Pipeline (detalle)

- NER
  - [“En”, “un”, “lugar”, “de”, **“La”, “Mancha”**, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]
  - [“En”, “un”, “lugar”, “de”, “**La_Mancha**”, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]

Pipeline (detalle)

- Mayúsculas / minúsculas
  - [“lugar”, “Mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]
  - [“lugar”, “mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]

```
for token in doc:
    if not(token.is_stop):
        print(token.lower_)
```

Pipeline (detalle)

- Lematización
  - [“lugar”, “mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]
  - [“lugar”, “mancha”, “cuyo”, “nombre , “querer”, “acordar yo”]

```
for token in doc:
    if not(token.is_stop):
        print(token.lemma_)
```

Recursos

Recursos lingüísticos disponibles

- Diccionarios
- Tesauros
- Lexicones

- Bases de datos lingüísticas
- Corpus

Diccionarios, Tesauros, Lexicones

- Diccionarios
  - RAE, Linguee, WordReference, TheFreeDictionary…
  - https://pypi.org/project/pyrae/
  - https://github.com/squat/drae
- Tesauros
  - TheFreeDictionary
- Lexicones
  - https://openlexicon.fr/datasets-info/

Bases de datos lingüísticas

- WordNet (https://wordnet.princeton.edu/)
- Spanish WordNet 3.0 (http://timm.ujaen.es/recursos/spanish-wordnet-3-0/)
- FrameNet (https://framenet.icsi.berkeley.edu/fndrupal/)

Corpus

- Un corpus es un conjunto de textos, que puede estar o no etiquetado, que se utiliza para tareas de procesamiento de leguaje natural
- Corpus de NLTK (https://www.nltk.org/nltk_data/)
- https://www.kaggle.com/datasets?tags=13204-NLP
- POS (https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1827)
- NER-C (https://www.kaggle.com/datasets/nltkdata/conll-corpora)
- MLDoc (https://github.com/facebookresearch/MLDoc)
- PAWS-X (https://github.com/google-research-datasets/paws/tree/master/pawsx)
- XNLI (https://github.com/facebookresearch/XNLI)

Herramientas y librerías

Herramientas y librerías

- NLTK
  - https://www.nltk.org

- Spacy
  - https://spacy.io

Para cerrar

Conceptos principales de esta sesión

- Normalizar es necesario para poder procesar
- El pipeline de normalización (básico)
  - Tokenizar -> quitar stopwords -> lematizar
- Recursos
  - Importante que sea posible acceder programáticamente
  - Corpus y Tesauros
- Herramientas y librerías
  - Spacy

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 3 – Etiquetado Morfosintáctico *(POS Tagging)*

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Encuesta previa

- ¿Sabes qué es un corpus lingüístico?
- ¿Sabes qué es un Modelo de Markov?
- ¿Has abierto ya un notebook con algo de código para NLP?

Problema del día

¿Cómo construir tablas de transición y emisión?

En el día de hoy

- Definición de moforsintaxis
- Categorías morfosintácticas
- Hidden Markov Models para POS tagging
- Probabilidad de transición / Probabilidad de emisión

Morfosintaxis / POS Tagging

morfosintaxis

1. f. *Ling.* Parte de la gramática que integra la morfología y la sintaxis.

Morfosintaxis / POS Tagging

- La morfosintaxis es un subcampo de la gramática que tiene como objetivo el estudio de la formación de las palabras y las relaciones de éstas para conformar frases u oraciones coherentes, es decir, con sentido lógico. (https://conceptodefinicion.de/morfosintaxis/)

- Parte de la gramática que integra la morfología y la sintaxis para identificar las partes de una oración (apuntes de la asignatura)

Morfosintaxis / **POS Tagging**

- Part of Speech:
  - In grammar, a part of speech or part-of-speech (abbreviated as POS or PoS, also known as word class or grammatical category) is a **category of words** (or, more generally, of lexical items) that have similar grammatical properties. (Wikipedia)

Categorías Morfosintácticas

Las categorías gramaticales variables

- Sustantivos
- Pronombres
- Determinantes
- Adjetivos
- Verbos

Las categorías gramaticales invariables

- Adverbios
- Preposiciones
- Conjunciones
- Interjecciones

Categorías Morfosintácticas

Sustantivos

- Poseen género y número (morfología)
- Forman sintagmas nominales (sintaxis)
- Designan entidades de diferente naturaleza (semántica)

  - Género / Número: Ingenier + a + s
  - Sintagmas Nominales: La ingeniera aeronáutica
  - Naturaleza (ejemplo): abstractos (Libertad) vs concretos (Mesa)

Categorías Morfosintácticas

Pronombres

- Sustituyen a los sustantivos y cumplen las mismas funciones
- Su significado concreto depende del **contexto** (p.ej. “ellos”)

Categorías Morfosintácticas

Determinantes

- Acompañan al sustantivo para precisar su significado
  - *El* coche
  - *Un* coche
  - *Ningún* coche
  - *Ese* coche

Categorías Morfosintácticas

Verbos

- Expresa una acción
- Es el núcleo del predicado
- Debe concordar en género y número con el sujeto
- Tiene variación de persona, número, tiempo, modo y aspecto

Etiquetado moforsintáctico

```
Bebo un vaso del vino tinto
```

Etiquetado moforsintáctico

Bebo un vaso del vino tinto

POS tagging informal

Bebo(verbo) un(determinante) vaso(sustantivo) de(determinante) el(pronombre) vino(sustantivo) tinto(adjetivo)

Etiquetado moforsintáctico

- Penn Treebank
  - Conjunto de etiquetas normalizadas
  - Diseñado para idioma inglés
  - https://gist.github.com/nlothian/9240750

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figura 1. Etiquetas para las categorías gramaticales en el Penn Treebank.
Fuente: Jurafsky y Martin, 2009.

Etiquetado moforsintáctico

Bebo un vaso del vino tinto

POS tagging informal

Bebo(verbo) un(determinante) vaso(sustantivo) de(determinante) el(pronombre) vino(sustantivo) tinto(adjetivo)

POS tagging con Penn Treebank

Bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ

Etiquetado moforsintáctico

- Penn Treebank
  - Ejemplo de datos etiquetados

Ejemplo de etiquetas


```
[
["Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.",["NNP","NNP","CD","NNS","JJ","MD","VB","DT","NN","IN","DT","JJ","NN","NNP","
["Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.",["NNP","NNP","VBZ","NN","IN","NNP","NNP","DT","NNP","VBG","NN"]] ,
["Rudolph Agnew, 55 years old and former chairman of Consolidated Gold Fields PLC, was named a nonexecutive director of this British industrial conglomerate.",["NNP
["A form of asbestos once used to make Kent cigarette filters has caused a high percentage of cancer deaths among a group of workers exposed to it more than 30 year
["The asbestos fiber, crocidolite, is unusually resilient once it enters the lungs, with even brief exposures to it causing symptoms that show up decades later, res
["Lorillard Inc., the unit of New York-based Loews Corp. that makes Kent cigarettes, stopped using crocidolite in its Micronite cigarette filters in 1956.",["NNP","
["Although preliminary findings were reported more than a year ago, the latest results appear in today's New England Journal of Medicine, a forum likely to bring ne
["A Lorillard spokewoman said, ``This is an old story.",["DT","NNP","NN","VBD","DT","VBZ","DT","JJ","NN"]] ,
["We're talking about years ago before anyone heard of asbestos having any questionable properties.",["PRP","VBG","IN","NNS","IN","IN","NN","VBD","IN","NN","VBG","D
["There is no asbestos in our products now.''",["EX","VBZ","DT","NN","IN","PRP$","NNS","RB"]] ,
```


https://github.com/nlp-compromise/penn-treebank/blob/master/penn-data.json

Etiquetado moforsintáctico

- Alternativas al Penn Treebank
  - EAGLES (https://www.cs.upc.edu/~nlp/tools/parole-sp.html)

Talamé, L., Cardoso, A., & Amor, M. (2019). Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python. In *XX Simposio Argentino de Inteligencia Artificial (ASAI 2019)-JAIIO 48 (Salta)*.

ADJETIVOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adjetivo	A
2	Tipo	Calificativo	Q
3	Grado	Apreciativo	A
4	Género	Masculino	M
		Femenino	F
		Común	C
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Caso	-	0
7	Función	Participio	P

Forma	Lema	Etiqueta
alegres	alegre	AQ0CP00
alegre	alegre	AQ0CS00
bonitas	bonito	AQ0FP00
bonita	bonito	AQ0FS00
bonitos	bonito	AQ0MP00
bonito	bonito	AQ0MS00
quemada	quemado	AQ0FS0P

Etiquetado moforsintáctico

- Demo con Spacy y NLTK

```python
import nltk
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize

# Texto de entrada
text_input = "I love this course about Artificial Intelligence"

# Tokenizar
list_tokens = word_tokenize(text_input)

# Obtener POS Tags
list_pos_tags = pos_tag(list_tokens)
for item in list_pos_tags:
  print(item)
```

```
('I', 'PRP')
('love', 'VBP')
('this', 'DT')
('course', 'NN')
('about', 'IN')
('Artificial', 'JJ')
('Intelligence', 'NN')
```

```python
import spacy

nlp = spacy.load("en_core_web_sm")

# Texto de entrada
text_input = "I love this course about Artificial Intelligence"
doc = nlp(text_input)

# Tokens y POS Tags
for token in doc:
    print(token.text, token.pos_, token.tag_)
```

```
I PRON PRP
love VERB VBP
this DET DT
course NOUN NN
about ADP IN
Artificial PROPN NNP
Intelligence PROPN NNP
```

Hidden Markov Models para POS Tagging

- Modelo oculto de Markov (HMM)
  - Un modelo oculto de Markov es un modelo estadístico que se puede representar como una **máquina de estados finitos**, pero donde las **transacciones entre estados son probabilísticas** y no determinísticas

Hidden Markov Models para POS Tagging

- Propiedad de Markov
  - la distribución de probabilidad del valor futuro de una variable aleatoria depende únicamente de su valor presente, siendo independiente de la historia de dicha variable.

$$P(X_{n+1}|X_n)$$

- Proceso de Markov
  - Secuencia de observaciones en las que cada observación cumple la propiedad de Markov

Hidden Markov Models para POS Tagging

- Propiedad de Markov
  - la distribución de probabilidad del valor futuro de una variable aleatoria depende únicamente de su valor presente, siendo independiente de la historia de dicha variable.

$$P(X_{n+1}|X_n)$$

- Proceso de Markov
  - Secuencia de observaciones en las que cada observación cumple la propiedad de Markov

¿Es una oración un proceso de Markov?
Es decir, ¿se cumple la propiedad de Markov?

Hidden Markov Models para POS Tagging

Modelo oculto de Markov

Modelo estadístico para determinar valores desconocidos (ocultos) a partir de valores observables en un proceso de Markov

Hidden Markov Models para POS Tagging

Modelo oculto de Markov

Modelo estadístico para determinar valores desconocidos (ocultos) a partir de valores observables en un proceso de Markov

Probabilidad de transición y de emisión

- Un HMM se compone de:
  - Conjunto de estados (POS tags)
  - Conjunto de observaciones (palabras | tags)
  - Matriz de **probabilidades de transición**
  - Matriz de **probabilidades de emisión**

Probabilidad de transición

- Las probabilidades de transición de etiqueta $P(ti|ti-1)$ representan la probabilidad de una etiqueta dada la etiqueta anterior.

```
Bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ
```

- Volviendo al ejemplo anterior, sería la probabilidad de que se tuviese un POS Tag (NN/Sustantivo) dado el POS Tag anterior (DT/Determinante)
- Se obtendría **contando** las veces que en un **corpus de trabajo** se tienen DT seguidos de NN, y dividiendo entre las veces que se tienen DT.

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

C(DT, NN) = 300
C(DT) = 1000
P(NN|DT) = 300/1000 = 0.3

Probabilidad de transición

- Las probabilidades de transición de etiqueta $P(ti|ti-1)$ representan la probabilidad de una etiqueta dada la etiqueta anterior.

- Matriz de **probabilidades de transición**

	PNN	MD	VB	JJ	NN	RB	DT
<S>	0.2767	0,0006	0,0031	0,0453	0,0449	0,0510	0.2026
PNN	0.3777	0,0110	0,0009	0,0084	0,0584	0,0090	0,0025
MD	0,0008	0,0002	0.7968	0,0005	0,0008	0.1698	0,0041
VB	0,0322	0,0005	0,0050	0,0837	0,0615	0,0514	0.2231
JJ	0,0366	0,0004	0,0001	0,0733	0.4509	0,0036	0,0036
NN	0,0096	0,0176	0,0014	0,0086	0.1216	0,0177	0,0068
RB	0,0068	0,0102	0.1011	0.1012	0,0120	0,0728	0,0479
DT	0.1147	0,0021	0,0002	0.2157	0.4744	0,0102	0,0017

Figura 4. Conjunto A de probabilidades de transición $P(t_i|t_{i-1})$ calculadas a partir del corpus WSJ.
Fuente: Jurafsky y Martin, 2009.

Se calcula a partir de un **corpus de trabajo**, y por tanto los valores dependen de dicho corpus

En la matriz aparecen los posibles estados, y es completamente **independiente de la oración** observada

Probabilidad de emisión

- Es probabilidad de que, dada una etiqueta, esta se asocie con una palabra concreta

```
Bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ
```

- Volviendo al ejemplo anterior, sería la probabilidad de que se tenga el POS Tag (NN/Sustantivo) dada la palabra “vino”
- Se obtendría **contando** las veces que en un **corpus de trabajo** se tienen la palabra “vino” junto con ese POS Tag.

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

C(NN, vino) = 150
C(NN) = 1500
P(vino|NN) = 150/1500 = 0.1

Probabilidad de emisión

- Es probabilidad de que, dada una etiqueta, esta se asocie con una palabra concreta

- Matriz de **probabilidades de emisión**

	Janet	will	back	the	bill
PNN	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figura 5. Conjunto B de probabilidades de observación calculadas a partir del corpus WSJ.
Fuente: Jurafsky y Martin, 2009.

Se calcula a partir de un **corpus de trabajo**, y por tanto los valores dependen de dicho corpus

En la matriz aparecen los posibles estados y **las palabras observadas**.

En resumen

- Etiquetado morfosintáctico o POS tagging
  - Asignar a cada palabra su categoría gramatical, de acuerdo al contexto concreto de la oración observada
- Hidden Markov Models para POS tagging
  - Modelo probabilístico en el que vamos a basar los algoritmos POS tagging
  - Requiere asumir cosas que no son del todo ciertas
- Probabilidad de transición / Probabilidad de emisión
  - Transición: dado un estado, probabilidad de cambiar a otro
  - Emisión: dado un estado observado, probabilidad de una observación concreta
  - Se calculan para un corpus dado

En la próxima sesión

¿Cómo identificar la función de una palabra en una oración con técnicas estadísticas?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python

Lorena Talamé, Alejandra Cardoso, Matías Amor[1]

[1] Universidad Católica de Salta,
Campo Castañares s/n, 4400, Salta

{mltalame, acardoso, matiasamor}@ucasal.edu.ar

Resumen. El procesamiento de lenguaje natural es un área central en cualquier aplicación de análisis de datos textuales. Muchas de estas aplicaciones se realizan a partir de textos publicados en Internet y muy frecuentemente, a partir de textos de redes sociales. Existen diversas herramientas informáticas de código libre que facilitan el procesamiento de lenguaje natural de diversas lenguas, pero especialmente para el idioma inglés. En este artículo se seleccionaron cinco módulos o librerías para el lenguaje Python, con el objetivo de comparar algunas funciones básicas del procesamiento de textos, aplicadas a textos cortos en español extraídos de una red social. La tokenización y segmentación de oraciones fueron perfectamente realizadas por dichas herramientas. Se encontraron diferencias en etiquetado gramatical pero los resultados se consideran aceptables.

Keywords: procesamiento de lenguaje natural, etiquetado gramatical, Python

1. Introducción

Con el uso de internet y redes sociales, compartimos información (multimedial o textual) que puede resultar valiosa en diversos ámbitos. Muchas personas utilizan las redes sociales para compartir su opinión referida a un evento, a un producto o a una situación en particular. Por lo tanto, resulta interesante poder obtenerlas y analizarlas automáticamente.

Por otro lado, la proporción de la población hispanohablante va en aumento. El español, en el mundo y como lengua materna, ocupa el segundo lugar, luego del chino mandarín [1]. Si nos centramos en los usuarios de Internet, las lenguas que más se emplean son el inglés, el chino y, en el tercer puesto, el español. Sin embargo, si se contempla que el chino, en general, lo usan los nativos, el español se situaría en segundo lugar tras el inglés.

El procesamiento de lenguaje natural (abreviado PLN o en sus siglas en inglés NLP, natural language processing) estudia la interacción entre una computadora y una persona facilitando la comunicación en lenguas naturales o humanas, como el idioma inglés o el español. En la actualidad, es un área de investigación y producción científica muy activa con numerosas aplicaciones.

Existen diversas herramientas informáticas que facilitan el procesamiento de texto en múltiples idiomas, especialmente para el idioma inglés. En este artículo se analizan algunas herramientas de código libre para el lenguaje Python, poniendo el acento especialmente en el análisis morfológico (determinar la forma o tipo gramatical de cada palabra o token en una oración) de textos cortos en lenguaje español, extraídos de la red social Twitter. El trabajo descrito en este artículo forma parte de un proyecto de mayor alcance que aún está en desarrollo.

2. Procesamiento de lenguaje

El procesamiento de lenguaje natural es un área de la inteligencia artificial que estudia la comunicación entre las personas y las computadoras utilizando lenguas naturales. Los lenguajes naturales son aquellos que utilizan los humanos para comunicarse ya sea en forma escrita u oral. Algunas aplicaciones típicas del PLN son la búsqueda de respuestas, corrección ortográfica, reconocimiento de voz, generación automática de resúmenes, traducción automática y análisis de sentimientos, entre otras. El proceso del PLN, generalmente, contempla cuatro tipos de análisis sobre la entrada de un texto en lenguaje natural (Fig. 1).

Fig. 1. Proceso PLN

A continuación, se describen los análisis indicados en la figura anterior:

- *análisis morfológico*: consiste en el análisis de las oraciones para identificar unidades léxicas o tokens (unidades del lenguaje con significación propia como palabras o signos de puntuación) y tipos de palabras (sustantivos, verbos, etc.). Una misma palabra puede adoptar diferentes roles en función del contexto en el que aparece, ocasionando problemas de ambigüedad. En el análisis morfológico se determina la forma, clase o categoría gramatical de cada palabra, es decir, se asigna a cada palabra de la oración, el rol que cumple dentro de ella. Esto se conoce como etiquetado gramatical o *POS* (Part-of-speech) *tagging*, el cual debe resolver las ambigüedades que presentan algunas palabras según el contexto donde se encuentren.

- *análisis sintáctico*: el objetivo de este análisis es determinar las relaciones de dependencia estructural entre las palabras de la oración estudiada. Esta dependencia y el orden de las palabras contribuyen a su significado. En la mayoría de las lenguas, este significado se transmite por la sintaxis. Por ejemplo, las dos oraciones: "*A ella le gusta Miguel de Cervantes*" y "*A Miguel de Cervantes le gusta ella*" difieren sólo en términos de sintaxis, pero transmiten significados diferentes. En este nivel se realiza

la desambiguación sintáctica de palabras que pueden funcionar como múltiples partes del discurso. La automatización de esta tarea se dificulta por la ambigüedad del lenguaje [2].

- *análisis semántico*: incluye la desambiguación de palabras con múltiples sentidos. Si se requiriera información del resto de la oración para esta desambiguación, se realizaría en este nivel. Se puede implementar una amplia gama de métodos para lograr la desambiguación, algunos de los cuales requieren información sobre la frecuencia con que ocurre cada sentido de las palabras, en un corpus particular de interés, y otros utilizan el conocimiento pragmático del dominio del documento en cuestión.

- *análisis pragmático*: es uno de los más complejos junto con el semántico. En muchas ocasiones las oraciones, y las palabras que las componen, no pueden ser interpretadas de manera literal ya que su significado depende en gran medida del contexto en el que se utilicen. Entonces, el análisis pragmático, a diferencia del sintáctico y semántico, va un paso más allá de la estructura del lenguaje, ya que es necesario entender el contexto donde las oraciones son emitidas para otorgarles significado.

Cabe mencionar que no siempre se aplican todos los niveles de análisis a cualquier tarea de PLN, depende del objetivo de la aplicación. En este trabajo nos enfocamos en el análisis morfológico.

Generalmente, el PLN necesita algunas acciones de “normalización” de los textos para poder “estructurarlos” y así analizarlos de una manera más fácil. A continuación, se definen algunos de estos procesos.

Tokenización

La tokenización es el proceso de dividir un documento de texto en sus distintos componentes, descartando los espacios en blancos y saltos de línea. Un token es una cadena de caracteres (palabra o un signo de puntuación) con algún significado en el contexto de un texto. Por ejemplo, el texto “El niño corre” tiene tres tokens: {'El', 'niño', 'corre'}.

Aunque la tokenización puede resultar trivial, no lo es si se considera que existen caracteres que algunas veces pueden ser parte de un token y otras veces pueden resultar ser separadores de tokens. Por ejemplo, el carácter punto puede ser parte de una abreviatura o sigla, pero también puede resultar ser separador de oraciones.

Segmentación

Consiste en separar el texto en fragmentos que puedan tratarse de forma independiente. La forma más usual de segmentación es dividir el texto en párrafos u oraciones.

Etiquetado gramatical

El etiquetado gramatical o *POS (part-of-speech) Tagging* consiste en la asignación de una etiqueta o categoría gramatical (sustantivo, verbo, adjetivo, etc.) a cada uno de los tokens de una oración según el rol que cumpla dentro de la misma. Por ejemplo,

para la oración "*El niño corre por la plaza*" el resultado del etiquetado POS se observa en la segunda fila de la Tabla 1.

Tabla 1. Ejemplo etiquetado POS

Token	**El**	**niño**	**corre**	**por**	**la**	**plaza**
Etiquetado POS	Articulo masculino singular	Nombre común masculino singular	Verbo indicativo presente 3ra. persona del singular	Preposición	Articulo femenino singular	Nombre común femenino singular

Los algoritmos de POS-tagging deben resolver las ambigüedades que presentan algunas palabras según el contexto donde se encuentren. Por ejemplo, en la oración "*El niño ayuda a su hermana*", la palabra "ayuda" se define como verbo; sin embargo, en la oración "*El niño necesita ayuda*" la misma palabra se establece como sustantivo.

Las categorías gramaticales resultan de gran utilidad por la información que proporcionan acerca de una palabra, sus palabras continuas y la forma de interpretarlas. Saber si una palabra es un sustantivo o un verbo dice mucho sobre las palabras junto a ellas y las formas de interpretarlas. Sirve además para encontrar entidades nombradas en textos y se aplica también a otras tareas de extracción de información.

Reconocimiento de entidades nombradas

El reconocimiento de entidades con nombres o entidades nombradas (*NER, Named Entity Recognition*) se ocupa de la extracción de unidades de información predefinidas, como nombres de personas, fechas, lugares, organizaciones, etc.

3. Herramientas informáticas para PLN

Existe una amplia variedad de herramientas informáticas para el PLN en diversos idiomas. En este trabajo se seleccionaron algunas Open Source que se pueden utilizar en programas Python[1]. La comunidad científica utiliza en mayor medida el lenguaje interpretado Python por lo que sólo nos centraremos en aquellas bibliotecas que se puedan utilizar en este lenguaje de programación. Cabe mencionar que, si bien no todas son nativas para este lenguaje, como Stanford NLP, existen plugins o APIs que permiten enlazarlas. En el contexto de este trabajo se utilizarán los términos "herramientas", "librerías", "bibliotecas" y "paquetes" como sinónimos.

[1] https://www.python.org

3.1. NLTK

NLTK[2] (Natural Language Toolkit) es una plataforma para la creación de programas Python para análisis de texto. Como fue creado con fines educativos en el año 2001, permite la realización de proyectos con distintos propósitos y alcance variado. Provee demostraciones paso a paso de distintos algoritmos [3]. Integra más de 50 corpus y un conjunto de librerías para segmentación, tokenización, etiquetado del habla, análisis sintáctico y semántico, entre otros. Para el español muchas de estas opciones no están disponibles.

Su instalación en Python es sencilla debido a que es un paquete que se encuentra en el repositorio del lenguaje. Con una simple instrucción, se instala lo necesario para utilizarlo.

El etiquetador POS de NLTK originalmente fue creado para el idioma inglés. Sin embargo, se lo puede extender al español entrenando el etiquetador con algún corpus que incorpora NLTK en esta lengua o usando un etiquetador externo como el de Stanford. Para este trabajo se seleccionó la primera opción con el corpus cess_esp [4] que tiene 500.000 palabras y 610 archivos.

El etiquetado POS que retorna NLTK se corresponde con el formato EAGLES[3]. Las etiquetas EAGLES codifican todas las características morfológicas existentes para la mayoría de los idiomas europeos, incluido el español. Estas etiquetas consisten en un conjunto de caracteres de longitud variable donde cada uno corresponde a una característica morfológica. El primer carácter en la etiqueta es siempre la categoría POS que, a su vez, determina la longitud y la interpretación de los elementos siguientes [5].

3.2. Freeling

Freeling[4] es una librería de código abierto para el análisis de texto (tokenización, análisis morfológico, detección de entidades nombradas, etiquetado POS, etc.) para una variedad de idiomas. Desde su primera versión, tiene soporte para el español. Fue desarrollado por el Centro de Tecnologías y Aplicaciones del Lenguaje y el Habla (TALP) de la Universidad Politécnica de Catalunya. Se puede utilizar y ampliar los recursos lingüísticos por defecto (diccionarios, lexicones, gramáticas, etc.) adaptándolos a dominios específicos.

Su código fue escrito en C++ bajo una arquitectura cliente – servidor. Existen diversas APIs que permiten utilizar Freeling en otros lenguajes, incluido Python. Su instalación no es sencilla, sobre todo en entornos Windows. Sin embargo, subsanado este inconveniente, al importar la API desde el programa Python se puede invocar a todas las funciones implementadas.

El analizador morfológico de Freeling realiza el etiquetado POS usando las etiquetas EAGLES.

---

[2] https://www.nltk.org/
[3] http://blade10.cs.upc.edu/freeling-old/doc/tagsets/tagset-es.html
[4] http://nlp.lsi.upc.edu/freeling/node/1

3.3. Pattern.es

Pattern.es[5] es una biblioteca para Python que contiene herramientas para la conjugación de verbos, la singularización o la pluralización de sustantivos, la división de chunks y permite realizar el etiquetado POS para español. En la versión para el idioma inglés adiciona una interfaz para WordNet. Su instalación y uso son muy sencillos.

Para el etiquetado gramatical utiliza Penn TreeBank Tags [6]. Penn TreeBank está basado en el corpus Brown, pionero en etiquetado POS para inglés. A pesar de haberse creado para el inglés, Pattern lo usa también para el español. Cada etiqueta está formada con dos o tres caracteres que indican la función que cumple cada palabra en una oración.

3.4. SPACY

SpaCy[6] es una biblioteca para el procesamiento avanzado de lenguaje natural en Python. Incluye modelos estadísticos pre-entrenados y vectores de palabras, admite tokenización para más de 45 idiomas. Provee etiquetado, análisis y reconocimiento de entidades nombradas y una fácil integración de aprendizaje profundo. Es muy sencilla de utilizar, basta con incorporar desde un programa Python el modelo para el español.

Devuelve un etiquetado POS completo, no sólo indica la función de la palabra en la oración, sino otros datos tales como tiempo verbal, persona, número, modo, género, entre otros. La tokenizacion y el etiquetado gramatical se basan en el corpus OntoNotes5[7] que sigue la sintaxis de Penn TreeBank.

3.5. Stanford NLP

Stanford NLP es un framework diseñado y desarrollado para Java que provee soluciones para la mayoría de las tareas más comunes en el procesamiento de lenguaje natural que van desde la tokenización hasta el reconocimiento del discurso [7]. No todas las funciones están implementadas para el español.

Existen APIs para otros lenguajes de programación, entre ellos, Python[8]. NLTK incluye algunas de sus funciones, aunque si se desea hacer un análisis más profundo, siempre se debe correr el servidor de Java en segundo plano y prescindir de NLTK.

Permite realizar etiquetado POS con el formato EAGLES y reconoce entidades para el análisis morfológico. Para el análisis sintáctico, identifica grupos lingüísticos o chunks y se puede obtener el árbol de dependencia.

Su instalación no es tan sencilla como otras de las herramientas analizadas por el hecho de tener que instalar el servidor Java y definir variables de entorno.

---

[5] https://www.clips.uantwerpen.be/pages/pattern-es
[6] https://spacy.io/
[7] https://catalog.ldc.upenn.edu/LDC2013T19
[8] https://stanfordnlp.github.io/stanfordnlp/

3.6. Comparación

De cada herramienta se tuvieron en cuenta algunos aspectos como la facilidad en su instalación, sobre todo para una persona con conocimientos básicos o nulos en programación; la necesidad de utilizar una interfaz de comunicación; la presencia de funciones específicas para segmentación y tokenización; el tipo de etiquetado POS que realizan y la implementación de lematizador para el lenguaje español.

A modo de resumen se presenta un cuadro comparativo de algunas funciones básicas de procesamiento de texto y las herramientas mencionadas que fueron objeto de evaluación (Tabla 2).

Tabla 2. Comparación de herramientas

Herramienta	Código	Segmentación y tokenización	Etiquetado POS	Lematización
NLTK	Nativo Python	Si	Eagles	No
Freeling	API	Si	Eagles	Si
Pattern.es	Nativo Python	Si	Penn TreeBank	Si
Spacy	Nativo Python	Si	Penn TreeBank	Si
Stanford NLP	API	Si	Eagles	Si

En la segunda columna de la Tabla 2 se muestra la relación de las herramientas con el lenguaje Python. NLTK, Pattern.es y Spacy se pueden instalar muy fácilmente de los repositorios de Python a través de la línea de comando. No ocurre lo mismo con Stanford NLP y Freeling. Ambos paquetes no son nativos de Python; fueron escritos en Java y C++, respectivamente. La instalación de Stanford es más sencilla que la de Freeling aunque ambos necesitan sendos servidores que se estén ejecutando para funcionar. Sin embargo, si no se requiere de todo el potencial de ambas herramientas, existe otra opción para su uso: Stanford se puede utilizar a través de NLTK y Freeling provee un ejecutable que realiza la segmentación, la tokenización, el etiquetado POS y la lematización recibiendo un archivo como entrada.

Todas las herramientas poseen funciones para segmentación y tokenización. NLTK, Freeling y Stanford NLP realizan el etiquetado POS con etiquetas Eagles mientras que Pattern.es y Spacy lo hacen con etiquetas de Penn TreeBank.

Salvo NLTK, las demás herramientas, implementan un lematizador para el idioma español. El lematizador disponible en NLTK, WordNet, está disponible para el idioma inglés.

4. Corpus

Para poder evaluar las diferentes alternativas mencionadas para el procesamiento del lenguaje natural, se diseñó un corpus de textos cortos obtenidos de la red social Twitter (llamados *tweets*), por lo cual los textos que serán objeto de estudio son subjetivos o coloquiales. Se capturaron tweets utilizando la API[9] de Twitter y a partir de una serie de hashtags sobre temas de actualidad de Argentina durante Octubre del año 2018. Los mensajes se almacenaron en una base de datos NoSQL. Se optó por trabajar con mensajes en lenguaje español, generados en nuestro país (según identificador de geolocalización) y se evitó mensajes con errores ortográficos. Si bien se recopilaron muchos tweets durante este mes, se descartaron una gran parte de ellos por contener solo imágenes, íconos o textos con poca información para el análisis posterior. Finalmente se seleccionaron al azar 100 mensajes, algunos de los ellos compuestos por más de una oración.

Como el objetivo de esta experimentación fue comparar las devoluciones de cada herramienta de PLN, en primer lugar, fue necesario realizar una serie de acciones para eliminar íconos, emojis, hashtags y links, los cuales son muy usuales en estos tipos de textos.

4.1. Preparación de documentos

Muchos comentarios en las redes sociales, no suelen tener en cuenta las reglas ortográficas, utilizan palabras del lunfardo o abreviaturas y muchas veces acentúan sus emociones con el uso de *emojis* (pequeñas imágenes que representan emociones, comidas, lugares, etc.), palabras completas en mayúsculas o signos de exclamación. Por esta razón, se realizaron una serie de acciones a modo de limpieza de los textos (Fig. 2) previo al análisis, ya que las herramientas analizadas sólo reconocen palabras válidas del idioma español.

Fig. 2. Preparación de documentos

Como primer paso, teniendo en cuenta que el objetivo del trabajo es comparar las herramientas a nivel textual, se eliminaron los emojis, url's y hashtags presentes en los tweets. Para esto se escribió un programa Python y se utilizaron expresiones regulares y la librería *emoji* que detecta símbolos Unicode.

La repetición de letras es usual para intensificar la emoción en un texto. Por ejemplo, "*Es taaaaaan pero tan difícil que yo me aleje de alguien que quería, pero si lo hice es*

[9] https://developer.twitter.com/

porque de verdad ya no aguantaba más.". En este caso, la palabra "*taaaaaan*" demuestra el acento que quiso poner el usuario, pero no es una palabra válida en español. Por esto, se descartaron las letras repetidas e innecesarias para tener una palabra reconocida en la lengua española [8]. En el ejemplo, se reemplazaron las seis letras "*a*" por una sola obteniendo como resultado "*tan*" que sí es correcta en esta lengua.

La última tarea realizada fue reemplazar aquellas abreviaciones que comúnmente se utilizan por la palabra completa. Por ejemplo, se reemplazó el "*x*" por "*por*" y "*xq*" por "*porque*", entre otras.

5. Experimentos

Para el estudio comparativo se evaluaron las funciones de segmentación de oraciones, tokenización y etiquetado POS en cada herramienta. Con el objeto de medir la capacidad de cada una y de tener un punto de comparación, se aplicaron, manualmente, éstas formas de procesamiento al corpus. Para facilitar las comparaciones posteriores se equipararon las etiquetas EAGLES y Penn TreeBank, definiendo nuevas etiquetas (segunda columna de Tabla 3) con las cuales se etiquetó el corpus. Las columnas siguientes de la tabla, contienen las etiquetas utilizadas por los analizadores (EAGLES y Penn TreeBank) y ejemplos. Nótese que de EAGLES solo se utilizaron los primeros caracteres que son los que contienen la información principal de cada etiqueta.

Tabla 3. Equivalencia de etiquetas

Descripción	Etiqueta	EAGLES	TreeBank	Ejemplo
Conjunción	CONJ	CC	CC	y, o
Determinante	DET	DA	DT	la, el
Adjetivo	ADJ	AQ	JJ	alegre, mala
Verbo	VERBO	VM, VA, VS	VB, VBZ, VBP	estoy, suman
Sustantivo	SUST	NC	NN, NNS	circo, noche
Nombre Propio	NPRO	NP	NP, NPS	Argentina
Adverbio	ADV	RG	ADP	siempre, pronto
Interjección	INTJ	I	IN	ah, ay
Signos Puntuación	PUNT	FAT, FP, FC	O	. ; !

5.1. Segmentación

El corpus está compuesto de 136 oraciones (algunos tweets contienen más de una oración) muchas de las cuales no estaban correctamente separadas dentro de cada tweet, es decir, luego de punto al final de oraciones no se encontró un espacio antes del comienzo de la oración siguiente, por ejemplo: "*Nosotros vivíamos en el campo.Nos relacionábamos con familias amigas*". Otro problema encontrado en el corpus es que muchos tweets después de un punto no comenzaban con mayúsculas. Un ejemplo de esto es "*Mi papá se jubila e invita a los vecinos a un costillar. tremenda felicidad*". Además, a otros comentarios les faltaba el punto para diferenciar una sentencia de otra. Para algunos textos, NLTK detectó más oraciones de las que realmente tienen. Estos son los casos que contienen más de un signo de puntuación consecutivo. Por ejemplo: *"¡¡¡Vacaciones!!! Lo bueno es que desperté y ya sabía que era domingo.*" Detectó tres oraciones: *"¡¡¡Vacaciones!!*", *"!*" y *"Lo bueno es que desperté y ya sabía que era domingo.* "

En primer lugar, se evaluó la capacidad de las herramientas para la segmentación. A pesar de los inconvenientes mencionados, todas las herramientas lograron distinguir las oraciones en un porcentaje aceptable, Stanford NLP logró el mejor resultado segmentando el 98,53% de las oraciones. En la segunda columna de la Tabla 4 se observa, la cantidad de oraciones detectadas por cada analizador y en la última columna los porcentajes correspondientes.

Tabla 4. Resultados segmentación

Oraciones en corpus	**136**	
Stanford NLP	134	98.53%
NLTK	129	94.85%
SPACY	124	91.18%
Freeling	123	90.44%
Pattern.es	128	94.17%

Para todo el análisis que continúa se solucionaron los problemas de separación de oraciones. De esta manera, dos herramientas lograron detectar la totalidad de sentencias: Stanford NLP y NLTK. Pattern sólo no encontró una. Freeling aumentó su rendimiento detectando el 95,58% de las oraciones. Spacy sólo encontró cuatro sentencias más que antes logrando un 94,12% de eficiencia. En la Tabla 5 se reflejan estos resultados.

5.2. Tokenización

Nltk, Spacy y Pattern no tokenizaron correctamente aquellas palabras precedidas por un signo de exclamación o interrogación. Por ejemplo, en la oración *"¡No creo en el circo de la información!"*, se consideró "*¡No*" como un único token cuando lo correcto son dos tokens: "*¡*" y "*No*".

En el corpus se identificaron 1481 tokens (palabras, signos de puntuación, números y fechas), de los cuales más del 90% fueron correctamente identificados por todas las herramientas.

Tabla 5. Resultados de segmentación con textos corregidos

Oraciones en corpus	**136**	
Stanford NLP	136	100%
NLTK	136	100%
SPACY	128	94.12%
Freeling	130	95.58%
Pattern.es	135	99.26%

5.3. Etiquetado POS

Las herramientas utilizan distintas nomenclaturas para el etiquetado POS pero, como se mencionó antes, se realizó una equiparación para permitir la comparación. En la Tabla 6 se observa el resultado de la evaluación de la oración *"¡No creo en el circo de la información!"*. En la segunda fila se observa el etiquetado realizado manualmente y en las filas posteriores las etiquetas devueltas por cada analizador (equiparadas según se describe en la Tabla 3). Obsérvese que "DESC" representa la etiqueta "None" que devuelve Nltk al no poder clasificar.

Tabla 6. Ejemplo etiquetado POS

Tweet	***¡***	***No***	***creo***	***en***	***el***	***circo***	***de***	***la***	***información***	***!***
	PUNT	**ADV**	**VERBO**	**PREP**	**DET**	**SUST**	**PREP**	**DET**	**SUST**	**PUNT**
Nltk	DESC		VERBO	PREP	DET	SUST	PREP	DET	SUST	PUNT
Freeling	PUNT	ADV	VERBO	PREP	DET	SUST	PREP	DET	SUST	PUNT
Pattern	NPRO		VERBO	INTJ	DET	SUST	INTJ	DET	SUST	PUNT
Spacy	SUST		VERBO	ADV	DET	SUST	ADV	DET	SUST	PUNT
Stanford	PUNT	ADV	VERBO	PREP	DET	SUST	PREP	DET	SUST	PUNT

A continuación, se describen algunas particularidades detectadas de cada librería.

NLTK

Una de las principales dificultades de esta herramienta es que no logra separar palabras precedidas o seguida de signos de puntuación (punto, exclamación o interrogación). Muchos de los casos analizados fueron etiquetados como "None", perdiendo así información muy útil sobre el etiquetado. Algunos ejemplos: "*cambiare!",* "*cariño?*".

FREELING

Si bien en este estudio no se realizó el reconocimiento de entidades nombradas es importante mencionar que Freeling las detecta de manera más acertada que el resto de las herramientas. Por ejemplo, en la oración "*La mejor noche con mis amigos en San Salvador de Jujuy*", Freeling indica que [San_Salvador_de_Jujuy] es un nombre propio (Tabla 7).

Tabla 7. Ejemplo 1

Tweet	***La***	***mejor***	***noche***	***con***	***amigos***	***en***	***San***	***Salvador***	***de***	***Jujuy***
Nltk	DT	ADJ	SUST	PREP	SUST	PREP	DESC	NPROP	PREP	DESC
Freeling	DT	ADJ	SUST	PREP	SUST	PREP	NPROP			
Pattern	DT	ADJ	SUST	PREP	SUST	INTJ	ADJ	NPROP	INTJ	NPROP
Spacy	DT	ADJ	SUST	ADV	SUST	ADV	NPROP	NPROP	ADV	NPROP
Stanford	DT	ADJ	SUST	PREP	SUST	PREP	NPROP	NPROP	PREP	NPROP

Solo algunas palabras (27) que comienzan con mayúsculas las clasifica como nombre propio cuando en realidad no lo son, por ejemplo "*Te Amo*". Este es un problema que encontramos en este tipo de textos, donde no siempre se respetan las reglas ortográficas. Freeling también identifica adverbios compuestos por más de una palabra, por ejemplo: "*por favor*" lo clasifica como adverbio (en este caso, adverbio de modo) mientras que las otras herramientas indican que son preposición y sustantivo (ejemplo en Tabla 8).

Tabla 8. Ejemplo 2

Tweet	***Por***	***favor***	***,***	***no***	***confundas***	***a***	***la***	***gente***
Nltk	PREP	SUST	PUNT	ADV	DESC	PREP	DT	SUST
Freeling	ADV		PUNT	ADV	VERBO	PREP	DT	SUST
Pattern	INTJ	ADV	PUNT	ADV	SUST	INTJ	DT	SUST
Spacy	ADV	SUST	PUNT	ADV	ADJ	ADV	DT	SUST
Stanford	PREP	SUST	PUNT	ADV	VERBO	PREP	DT	SUST

PATTERN

En el corpus existen cuatro menciones a días de la semana (ejemplo "*Primera vez que no quiero que llegue el viernes*"). Pattern los clasifica como nombres comunes o sustantivos, sin embargo, las otras herramientas les asignan la etiqueta "fecha".

Al igual que NLTK, no detecta los signos de exclamación e interrogación de apertura. A las preposiciones las identifica como interjecciones.

SPACY

A muchas de las preposiciones las clasifica como adverbios y las fechas como sustantivo. Sin embargo, si se incluyen otros argumentos al análisis, se observa que las identifica correctamente, es decir devuelve adtype=prep y advtype= time, respectivamente.

STANFORD

Tanto Stanford cono Freeling, a las contracciones las dividen en dos palabras. Por ejemplo, la contracción "del" la separa en "de" y en "el" asignándoles las etiquetas preposición y determinante, respectivamente.

De los diez números existentes en el corpus, solo identifica nueve. Las palabras "sábado" y "viernes" se clasifican como sustantivo, a diferencia de Freeling que les asigna la etiqueta "fecha".

6. Resultados

Freeling etiquetó correcta y completamente los tweets, en el 28% de los casos y Stanford en el 19%, el resto de las herramientas no superaron el 6%, y esto se debe, a que en muchos textos Spacy, Nltk y Pattern no diferencian los tokens que comienzan con signos de puntuación. Se observa que modificando los textos para una correcta tokenización, los porcentajes de estas últimas herramientas mejoran, pero no superan

los valores obtenidos por Freeling y Stanford. El resto de los textos tuvieron coincidencia en gran parte del etiquetado, así, todas las herramientas coincidieron con el etiquetado manual entre el 64% y 75% (Tabla 9).

Tabla 9. Resultados evaluación

	Nltk	**Freeling**	**Pattern**	**Spacy**	**Santford**
Total	3%	28%	4%	6%	19%
Parcial	72%	64%	64%	75%	72%

7. Conclusiones

El procesamiento de lenguaje natural es un área central en cualquier aplicación de análisis de datos textuales. Muchas de estas aplicaciones se realizan a partir de textos publicados en Internet y muy frecuentemente, a partir de textos de redes sociales. Por otro lado, la variedad de herramientas Open Source para procesamiento de textos es amplia, lo cual facilita cualquier tipo de investigación en este campo. En este artículo se utilizaron cinco herramientas o librerías para evaluar sus resultados sobre un corpus de textos cortos, en español y con términos coloquiales.

A diferencia de otros trabajos que realizaron comparaciones de herramientas de PLN, para la creación y explotación de corpus [9] se utilizaron textos escolares o fuentes formales (por ejemplo en [10] se construyó un corpus de sentencias), el corpus de textos de este trabajo fue obtenido de una red social. Los comentarios en redes sociales suelen contener imágenes, links, íconos, emojis, y muchas veces contenido textual incorrecto desde el punto de vista ortográfico. Por esta característica fue necesaria una etapa de preparación previa al análisis de las herramientas.

Se compararon tres funciones básicas del PLN: segmentación de oraciones, tokenización y etiquetado gramatical. Respecto a los dos primeros no se observaron mayores diferencias. Considerando el etiquetado completo y correcto de cada oración, Freeling tuvo la mayor cantidad de aciertos.

Desde el punto de vista de la sencillez de instalación, las herramientas nativas de Python permiten utilizar repositorios lo que facilita su uso a personas no expertas en programación. Tanto Stanford como Freeling, necesitan conocimiento más específico para utilizar sendos servidores.

El resultado de esta comparación será utilizado en otro proyecto que está desarrollando este equipo de investigación. Aunque las herramientas analizadas proveen más opciones para el análisis de textos, nuestro objetivo consistió en evaluar sólo aquellas que serán utilizadas en el proyecto principal.

Referencias

[1] Instituto Cervantes, «El español: una lengua viva. Informe 2018,» 2018. [En línea]. Available: https://cvc.cervantes.es/lengua/espanol_lengua_viva/pdf/espanol_lengua_viva_2018.pdf. [Último acceso: 02 2019].

[2] S. N. Galia Haro y A. Gelbukh, Investigaciones en análisis sintáctico para el español, México: Instituto Politécnico Nacional, Dirección de Publicaciones, 2007.

[3] S. Bird y E. Loper, «NLTK: The Natural Language Toolkit,» *ACL Association for Computacional Linguistics,* 2002.

[4] M. Martí, M. Taulé, L. Márquez y M. Bertran, «Cess-ece: corpus anotados del español y catalán.,» *Arena Romanistica. A New Nordic Journal of Romance Studies,* vol. 1, 2007.

[5] Research Center on Language and Speech Technologies and Applications, «Freeling 4.1,» [En línea]. Available: https://talp-upc.gitbook.io/freeling-4-1-user-manual/tagsets. [Último acceso: 02 2019].

[6] «Penn Treebank P.O.S. Tags,» [En línea]. Available: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html. [Último acceso: 02 2019].

[7] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard y D. McClosky, «The Stanford Core Natural Language Processing Toolkit,» *Association of Computer Linguistics,* pp. 55-60, 2014.

[8] S. Brody y N. Diakopoulos, «Coooooooooooooollllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs,» *Empur. Methods Nat. Lang. Process.,* pp. 562-570, 2011.

[9] J. Vivaldi, «Catálogo de herramientas informátivas relacionadas con la creación, gestión y explotación de corpus textuales,» *Revista Tradumàtica - Traducció i Tecnologies de la Informació i la Comunicació,* nº 07, p. http://www.fti.uab.cat/tradumatica/revista, 2009.

[10] B. Valverde Olmedo y R. Martinez García, «Herramientas de código abierto para el procesamiento del español. Análisis y comparativa,» *Dialnet. Revista hiberoamericana de lingüística,* nº 11, pp. 287-308, 2016.

[11] E. D. Liddy, Natural Language Processing, 2001.

[12] B. Felbo, A. Mislove, A. Sogaard, I. Rahwan y S. Lehmann, Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm, 2017.

[13] M. P. Marcus, M. A. Marcinkiewicz y B. Santorini, «Building a Large Annotated Corpus of English: The Penn Treebank,» *Journal Computational Linguistics,* vol. 19, pp. 313-330, 1993.

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 3 – Etiquetado Morfosintáctico *(POS Tagging)*

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Problema del día

¿Cómo identificar la función de una palabra en una oración con técnicas estadísticas?

Encuesta previa

- Enuncia la definición de probabilidad de transición
- Enuncia la definición de probabilidad de emisión
- ¿En cuál de las dos influye el corpus lingüístico con el que se trabaja?

En el día de hoy

- Repaso de la sesión anterior
- Algoritmo de Viterbi
- Etiquetado basado en Machine Learning
- Named-Entity Recognition

Repaso

- Nuestro objetivo: POS tagging

```
Bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ
```

- Nuestro modelo: Hidden Markov Models

- Nuestros ingredientes:
  - Un conjunto de estados posibles (etiquetas)
  - Una observación (oración a etiquetar)
  - Matriz de probabilidades de transición
  - Matriz de probabilidades de emisión

Repaso

- Cosas que hemos asumido como ciertas:
  - La probabilidad de aparición de una **palabra** depende solo de **su propia etiqueta** y es independiente de las palabras y etiquetas vecinas.
  - La probabilidad de una **etiqueta** solo depende de la **etiqueta anterior**, en lugar de toda la secuencia de etiquetas

Repaso

- Cosas que hemos asumido como ciertas:
  - La probabilidad de aparición de una **palabra** depende solo de **su propia etiqueta** y es independiente de las palabras y etiquetas vecinas.

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

  - La probabilidad de una **etiqueta** solo depende de la **etiqueta anterior**, en lugar de toda la secuencia de etiquetas

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

Algoritmo de Viterbi: objetivo

- ¿Cuáles son las categorías gramaticales de cada palabra en la siguiente oración?

El señor vino tarde

Otra formulación, más ajustada a la computación:

- ¿Cuál es la secuencia de etiquetas más probable para la secuencia de palabras observada?

Algoritmo de Viterbi: observación

- El
- Señor
- Vino
- Tarde

Algoritmo de Viterbi: conjunto de etiquetas

- El
  - Artículo Determinado
- Señor
  - Sustantivo común
  - Sustantivo propio
  - Adjetivo
- Vino
  - Sustantivo común
  - Verbo
- Tarde
  - Sustantivo común
  - Adverbio
  - Verbo

- Artículo Determinado
- Sustantivo común
- Sustantivo propio
- Adjetivo
- Verbo
- Adverbio

https://www.buscapalabra.com/categoria-gramatical-tiempo-verbal.html

Algoritmo de Viterbi: matriz de probabilidad de transición

Valores del WSJ Corpus

	DT	NN	NNP	JJ	VBD	RB
DT	0,0017	0,4744	0,1147	0,2157	0,0002	0,0102
NN	0,0068	0,1216	0,0096	0,0086	0,0314	0,0177
NNP	0,0025	0,0584	0,3777	0,0084	0,0009	0,0090
JJ	0,0036	0,4509	0,0366	0,733	0,0001	0,0036
VBD	0,2231	0,0615	0,0322	0,0837	0,0050	0,0514
RB	0,0479	0,0120	0,0068	0,1012	0,1011	0,0728
<start>	0,2026	0,0449	0,2767	0,0453	0,0031	0,0510

Algoritmo de Viterbi: matriz de probabilidad de emisión

(valores ficticios, en un caso real son mucho más pequeños)

	El	Señor	Vino	Tarde
DT	0,5	0	0	0
NN	0	0,2	0,1	0,05
NNP	0	0,15	0	0
JJ	0	0,1	0	0
VBD	0	0	0,4	0,2
RB	0	0	0	0,4

Algoritmo de Viterbi: valores de viterbi

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t)$$

	<start>	El	Señor	Vino	Tarde	<end>
<end>						
DT						
NN						
NNP						
JJ						
VBD						
RB						
<start>	1					

- *Máximo de: valor de viterbi previo * p.transición * p.emisión*

Algoritmo de Viterbi: valores de viterbi

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0				
DT		0,1013				
NN		0				
NNP		0				
JJ		0				
VBD		0				
RB		0				
<start>	1	0				

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t)$$

- *Máximo de: valor de viterbi previo * p.transición * p.emisión*

- V[1,1] = max([V[0,7]*P(DT|<start>)*P(El|DT)]) = max([1*0.2026*0.5]) = 01013
- V[1,2] = max([V[0,7]*P(NN|<start>)*P(El|NN)]) = max([1*0.0449*0]) = 0
- …

Algoritmo de Viterbi: valores de viterbi

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0			
DT		0,1013	0			
NN		0	0,00961134			
NNP		0	0,00174287			
JJ		0	0,00218504			
VBD		0	0			
RB		0	0			
<start>	1	0	0			

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t)$$

- *Máximo de: valor de viterbi previo * p.transición * p.emisión*

- V[2,2] = max([V[1,7]*P(NN|DT)*P(Señor|NN)]) = max([0.1013*0.4744*0.2]) = 0.00961134
- …

Algoritmo de Viterbi: ruta de máxima probabilidad

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0		
DT		0,1013	0	0		
NN		0	0,00961134	0,00011687		
NNP		0	0,00174287	0		
JJ		0	0,00218504	0		
VBD		0	0	0,00012072		
RB		0	0	0		
<start>	1	0	0	0		

- Cuando hay más de una ruta de Viterbi posible, se elige la que de mayor probabilidad.

Algoritmo de Viterbi: ruta de máxima probabilidad

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0	0	
DT		0,1013	0	0	0	
NN		0	0,00961134	0,00011687	7,1059E-07	
NNP		0	0,00174287	0	0	
JJ		0	0,00218504	0	0	
VBD		0	0	0,00012072	7,3397E-07	
RB		0	0	0	2,482E-06	
<start>	1	0	0	0	0	

El/DT señor/NN vino/VBD tarde/RB

Etiquetado basado en Machine Learning

- Diferencia con respecto a otros ámbitos (señal, imagen, datos, etc)

¿Cómo representamos el texto a la entrada?

Palabras → Vectores

El señor come pescado

Como categorías (one-hot)

El	señor	Come	Pescado
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Bag of Word

- Recuento
- Frecuencia relativa
- TF-IDF

Word embeddings

- Word2Vec
- Glove

Métodos pre-entrenados que producen vectores de menor dimensionalidad (aprox. 300)

Etiquetado basado en Machine Learning

- Diferencia con respecto a otros ámbitos (señal, imagen, datos, etc)

¿Cómo representamos la secuencialidad?

Etiquetado basado en Machine Learning

Chiche, A., Yitagesu, B. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *J Big Data* **9**, 10 (2022). https://doi.org/10.1186/s40537-022-00561-y

Named Entity Recognition

- Definición

Reconocimiento de entidades con nombre (NER) es una de las características que ofrece Azure Cognitive Service for Language, una colección de algoritmos de aprendizaje automático e inteligencia artificial en la nube para el desarrollo de aplicaciones inteligentes en las que interviene lenguaje escrito. La característica Reconocimiento de entidades con nombre puede identificar y clasificar entidades en texto no estructurado. Por ejemplo: personas, lugares, organizaciones y cantidades. (Microsoft)

Busca localizar y clasificar en categorías ***predefinidas****, como personas, organizaciones, lugares, expresiones de tiempo y cantidades, las entidades nombradas encontradas en un texto. (Wikipedia)*

Named Entity Recognition

- Categorías predefinidas
  - **PER**: personas, como por ejemplo el nombre de alguien (ej.: Frodo Baggins)
  - **GPE**: ubicaciones geopolíticas: países, ciudades o estados (ej.: Madrid).
  - **LOC**: ubicaciones concretas no geo-políticas (ej.: Vesubio).
  - **ORG**: organizaciones o empresas (ej.: Microsoft).
  - **MONEY**: referencias a dinero (ej.: $ 5).
  - **DATE:** fechas (ej.: 05/02/2021 o «jueves»).
- Categorías específicas
  - P.ej: book-name; weapon; hardware;

Named Entity Recognition

- Ejemplo
  - “En un lugar de La Mancha, de cuyo nombre no quiero acordarme…”
  - “Yo he visto cosas que vosotros no creeríais. Atacar naves en llamas más allá de Orión. He visto rayos-C brillar en la oscuridad cerca de la Puerta de Tannhäuser. Todos esos momentos se perderán en el tiempo, como lágrimas en la lluvia. Es hora de morir.”

Named Entity Recognition

- Técnicas
  - Goyal, A., Gupta, V., & Kumar, M. (2018). Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, *29*, 21-43.
  - Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 50-70.
- Sistemas basados en reglas
- Aprendizaje automático

En resumen

- Algoritmo de Viterbi
  - Cálculo de los valores de Viterbi, y luego vuelta atrás
  - Depende del corpus que ha dado lugar a las prob. de transmisión y emisión
  - Se hacen suposiciones para simplificar

- POS Basado en machine learning
  - Reto: representar adecuadamente el texto a la entrada de la red
  - Apendizaje supervisado

- Named-Entity Recognition
  - Identificar fragmentos de texto como categorías pre-establecidas

En la próxima sesión

¿Cómo construir el árbol sintáctico de una oración?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 4 – Análisis Sintáctico

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Problema del día

¿Cómo construir el árbol sintáctico de una oración?

(y Presentación de la Actividad 1)

Encuesta previa

¿Qué tal vais con la entrega de la actividad individual?

¿Sabes qué es el árbol sintáctico de una oración?

En el día de hoy

- Sintaxis, análisis sintáctico
- Gramáticas de estructura sintagmática
- Métodos para el análisis sintáctico basado en programación dinámica

De dónde venimos…

El señor vino tarde

- Análisis morfológico
  - el Definite=Def|Gender=Masc|Number=Sing|PronType=Art
  - señor Gender=Masc|Number=Sing
  - venir Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin
  - tarde

- Análisis morfosintáctico (POS)
  - DET
  - NOUN
  - VERB
  - ADV

A dónde vamos

https://nlp.lsi.upc.edu/freeling/demo/demo.php

A dónde vamos

El análisis sintáctico

- En el análisis sintáctico se determinan las relaciones estructurales entre palabras (apuntes de la asignatura)

- El análisis sintáctico es, en el campo de la lingüística, el análisis de las **funciones sintácticas** o relaciones de **concordancia** y **jerarquía** que guardan las palabras agrupándose entre sí en **sintagmas** u **oraciones**. (www.wikilengua.org)

El árbol sintáctico

- Oración: conjunto de sintagmas
- Sintagma (o grupo): constituyente sintáctico formado por dos o más constituyentes sintácticos
- Nodos: constituyentes sintácticos
- Hojas: palabras de la oración

Sintagma Nominal (SN)	Sustantivo Pronombre	Sujeto. Atributo. Complemento directo. Complemento indirecto (solo si el núcleo es un pronombre). Complemento predicativo. Complemento circunstancial.
Sintagma Verbal (SV)	Verbo	Predicado (predicado verbal o predicado nominal).

Gramáticas de estructura sintagmática

- Formalizan, con el objetivo de llevar a cabo tratamiento computacional, el conocimiento de la gramática.

  - Vocabulario terminal **T**
  - Vocabulario no terminal **N**
    - **V** es la unión entre **T** y **N**

  - Conjunto de reglas **R**
  - Símbolo inicial **I**

Gramáticas de estructura sintagmática

- Ejemplo:

- **T** = {j,a}
- **N** = {J, A, E}
- **R** = {
  - E→ JA
  - J → j
  - A → a
  - A → Aa
  - E → EE

  }
- **I** = E

- jajajaa
  - **jajajaa**
  - Jajajaa
  - JAjajaa
  - Ejajaa
  - EJajaa
  - EJAjaa
  - EEjaa
  - EEJaa
  - EEJAa
  - EEJA
  - EEE
  - EE
  - E

Relaciones estructurales

Son reglas para la formación de sintagmas, o de oraciones

- Ejemplos:

SN → Det N

Se lee: El **sintagma nominal** se puede formar con un **determinante** y un **nombre**, en ese orden.

O → SN SV

Se lee: Una oración se puede formar con un **sintagma nominal** seguido de un **sintagma verbal**, en ese orden.

Gramáticas de estructura sintagmática

T = {la, una, cultura, literatura, novedad, árabe, griega, moderna}
N = {D,N,A,GA,GN,SN}
R = {D → {la, una}
N → {cultura, literatura, novedad}
A → {árabe, griega, moderna}
GA → A
GA → A GA
GN → N GA
SN → D GN}
I = SN

- una cultura → D N
- la literatura moderna → D N A → D N GA → D GN → SN

Tipos de gramáticas

- De estados finitos
  - En el lado izquierdo, símbolo NO TERMINAL
  - En el lado derecho ≥ 1 , por lo menos uno debe ser terminal
- **Libres de contexto**
  - En el lado izquierdo, símbolo NO TERMINAL
  - En el lado derecho ≥ 0 , terminales o no terminales
- Sensibles al contexto
  - Número de símbolos izda. $\leq$ Número de símbolos dcha.
- Enumerables recursivamente
  - Sin restricciones

Tipos de gramáticas

- Con categorías complejas

T ={el, la, los, las, un, uno, una, unas, unos, señor, señora, gato, gata, ciudad, vino, viajó, comió, tarde, pronto}

N = {SN, SV, DET, N, Vt, Vi, CC}

R =

- DET → {el, la, los, las, un, uno, una, unas, unos}
- N → {señor, señora, gato, gata, ciudad}
- Vt → {comió}
- Vi → {viajó, vino}
- CC → {tarde, pronto}
- O → SN SV
- SN → N
- SN → DET N
- SV → Vi CC
- SV → Vi
- SV → Vt SN

I = O

La señor vino tarde

El señores vino tarde

El gata vino tarde

Tipos de gramáticas

- Con categorías complejas

T ={el, la, los, las, un, uno, una, unas, unos, señor, señora, gato, gata, ciudad, vino, viajó, comió, tarde, pronto}

N = {SN, SV, DETms, DETmp, DETfs, DETfp, Nms, Nfs, Vt, Vi, CC}

I = O

R =

- **DETms → {el, un, uno}**
- **DETfs → {la, una}**
- **DETmp → {los, unos}**
- **DETfp → {las,unas}**
- **Nms → {señor, gato}**
- **Nfs → {señora, gata, ciudad}**
- Vt → {comió}
- Vi → {viajó, vino}
- CC → {tarde, pronto}
- O → SN SV
- SN → N
- **SN → DETms Nms**
- **SN → DETfs Nfs**
- SV → Vi CC
- SV → Vi
- SV → Vt SN

Tipos de gramáticas

- Con categorías complejas

El = {DET, m, s}

La = {DET, f, s}

…

Señor = {N, m, s}

…

N = {SN, SV, DET, N, Vt, Vi, CC}

I = O

R =

- O → SN [num= NUM, gen=GEN] SV [num= NUM, gen=GEN]
- SN [num= NUM, gen=GEN] → N [num= NUM, gen=GEN]
- **SN** [num= NUM, gen=GEN] →

  DET [num= NUM, gen=GEN] N [num= NUM, gen=GEN]
- …

Tipos de gramáticas

- **Libres de contexto**
  - En el lado izquierdo, símbolo NO TERMINAL
  - En el lado derecho ≥ 0 , terminales o no terminales

- **Forma normal de Chomsky**
  - Una gramática formal está en **Forma normal de Chomsky** si todas sus reglas de producción son de alguna de las siguientes formas
    - A → a
    - A → BC
  - Cualquier gramática libre de contexto se puede transformar en su CNF

Algoritmo Cocke-Kasami-Younger (CKY)

- Conceptos básicos
  - Determina si una secuencia pertenece al lenguaje definido en una GLC
  - Requiere que la GLC esté en forma normal de Chomsky
    - Toda GLC se puede escribir en forma normal
  - Destaca por su eficiencia

Algoritmo Cocke-Kasami-Younger (CKY)

- Pasos

1. Transformar la gramática a su forma normal de Chomsky
2. Construir una matriz triangular a partir de la oración
3. Rellenar la matriz
4. Reconstruir el árbol sintáctico

Algoritmo Cocke-Kasami-Younger (CKY)

1. Transformar la gramática a su forma normal de Chomsky

Algoritmo Cocke-Kasami-Younger (CKY)

1. Transformar la gramática a su forma normal de Chomsky

Lexicón:

N = {gato, pescado, perro, pez, ..}

Det = {el, la, una,…}

Vi = {salta, duerme, …}

Vt = { ve, ama, come,…}

Reglas:

1) O -> SN SV

2) SN -> N

3) SN -> Det N

4) SV -> Vi

5) SV -> Vt SN

Gramática en CNF

1) O -> SN SV

2) SN -> gato | pescado | perro | pez

3) Det -> el | la | una

4) SN -> Det N

5) N -> gato | pescado | perro | pez

9) SV -> salta | duerme

10) Vt -> ve | ama | come

11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

2.Construir una matriz triangular a partir de la oración

El	gato	come	pescado
(0,1)	(0,2)	(0,3)	(0,4)
	(1,2)	(1,3)	(1,4)
		(2,3)	(2,4)
			(3,4)

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1)	(0,2)	(0,3)	(0,4)
	(1,2)	(1,3)	(1,4)
		(2,3)	(2,4)
			(3,4)

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2)	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4) SV
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4) O
		(2,3) Vt	(2,4) SV
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4) O
	(1,2) SN N	(1,3)	(1,4) O
		(2,3) Vt	(2,4) SV
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

4. Reconstruir el árbol

Algoritmo Cocke-Kasami-Younger (CKY)

Ambigüedad

"Book the flight through Houston"

Algoritmo CKY probabilístico

- Gramática con probabilidades
- El cálculo de las celdas arrastra la probabilidad

El	gato	come	pescado
(0,1) Det 0.55	(0,2) SN 0.077	(0,3)	(0,4) O 0.00111
	(1,2) N 0.2 SN 0.2	(1,3)	(1,4) O 0.00288
		(2,3) Vt 0.2	(2,4) SV 0.013
			(3,4) SN 0.15 N 0.15

0.7*0.55*0.2

Gramática en CNF

O -> SN SV (0.8)
SN -> gato (0.2)
SN -> pescado (0.15)
Det -> el (0.55)
SN -> Det N (0.7)
N -> gato (0.2)
N -> pescado (0.15)
SV -> salta (0.35)
SV -> duerme (0.27)
Vt -> ve (0.7)
Vt -> ama (0.1)
Vt -> come (0.2)
SV -> Vt SN (0.6)

En la próxima sesión

¿Cómo representar la semántica de las palabras y realizar razonamientos?

Ojo! Actividad entregable

- Trabajo: caracterización de textos
  - Tareas básicas con Spacy
  - Entrega el **06 de mayo**
  - Ejercicio de código fuente y revisión de API

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 5 – Análisis Semántico

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo representar la semántica de las palabras y realizar razonamientos?

Encuesta previa

- ¿Conoces DBPedia?

- (P1) Si llueve, la calle se moja.

  (P2) No llovió.

  (C) La calle no está mojada.

  - ¿Te suena?

- ¿Conoces la Web Semántica?

En el día de hoy

- Utilidad del análisis semántico
- Representación formal del significado
  - Representación formal con lógicas descriptivas
  - Representación formal con lógicas de primer orden
- Gramáticas Libres de Contexto con anotaciones semánticas
  - Análisis semántico dirigido por la sintaxis

Utilidad del análisis semántico

- Convertir el conocimiento contenido en un texto en un modelo formal con el que poder realizar operaciones lógicas

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ex felis, fermentum eu cursus a, semper nec est. Morbi porttitor.

Representación del conocimiento

- Debe ser verificable, inequívoca, expresiva y permitir la inferencia de nuevo conocimiento.

- Algunas representaciones
  - Lógica de primer orden
    - Lógicas descriptivas
  - Redes Semánticas
  - Frames

Lógicas descriptivas

- Subconjunto de la expresividad de la lógica de primer orden

- Mecanismos para:
  - generar descripciones de conceptos
  - introducir axiomas
  - introducir propiedades de los individuos
  - Inferir nuevos conocimientos a partir de los axiomas

Formalismo de las lógicas descriptivas

- No hay un formalismo único, el término lógica descriptiva denota a una familia de formalismos.
- El más utilizado es ALC.
- De las lógicas descriptivas derivan OWL y la Web Semántica

Formalismo de tripletas

- Ibañez es Autor
- Ibañez escribió El sulfato atómico
- Los autores son personas

Cormenzana, Roberto & López-Borrull, Alexandre. (2018). ESTUDIO DE LA ADAPTACIÓN A RDA Y BIBFRAME EN EL ÁMBITO DE LAS BIBLIOTECAS ESPAÑOLAS: ESTUDIOS DE CASO Title: STUDY OF THE ADAPTATION TO RDA AND BIBFRAME IN THE FIELD OF SPANISH LIBRARIES: CASE STUDIES. Anales de Documentacion. 21. 10.6018/analesdoc.21.2.323441.

Operaciones con lógicas descriptivas

- Subsunción
  - Determinar si una categoría es un subconjunto de otra
- Clasificación
  - Determinar si un individuo pertenece a una categoría

Operaciones con lógicas descriptivas

- Subsunción
  - Determinar si una categoría es un subconjunto de otra

- Clasificación
  - Determinar si un individuo pertenece a una categoría

Lógica de primer orden

- Es un sistema formal diseñado para estudiar la inferencia en los lenguajes de primer orden. (*Wikipedia*)

- Recogemos el **significado** de una oración en lenguaje natural y lo representamos de manera formal, para poder hacer operaciones matemático/lógicas.

- El formalismo es fijo, las variables y funciones son **arbitrarias**.

Ejemplos de la lógica de primer orden

- Alberto es profesor
  - Profesor(Alberto)

- Alberto es profesor de Jorge
  - ProfesorDe(Alberto, Jorge)
  - ProfesorDe(Alberto, Jorge) ^ ProfesorDe(Alberto, Juan)

- Los alumnos son personas
  - $\forall$ (x) Alumno(x) $\rightarrow$ Persona(x)

- Existe un profesor que es una persona
  - $\exists$ (x) Profesor(x) $\rightarrow$ Persona(x)

Formalismo de la lógica de primer orden

$$
\begin{array}{rcl}
\textit{Sentencia} & \rightarrow & \textit{SentenciaAtómica} \\
 & | & (\textit{Sentencia Conectiva Sentencia}) \\
 & | & \textit{Cuantificador Variable}\ldots\ \textit{Sentencia} \\
 & | & \neg\textit{Sentencia} \\
 & & \\
\textit{SentenciaAtómica} & \rightarrow & \textit{Predicado}(\textit{Término}\ldots) \mid \textit{Término} = \textit{Término} \\
 & & \\
\textit{Término} & \rightarrow & \textit{Función}(\textit{Término}\ldots) \\
 & | & \textit{Constante} \\
 & | & \textit{Variable} \\
 & & \\
\textit{Conectiva} & \rightarrow & \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
\textit{Cuantificador} & \rightarrow & \forall \mid \exists \\
\textit{Constante} & \rightarrow & A \mid X_1 \mid \textit{Juan} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicado} & \rightarrow & \textit{AntesDe} \mid \textit{TieneColor} \mid \textit{EstáLLoviendo} \mid \cdots \\
\textit{Función} & \rightarrow & \textit{Madre} \mid \textit{PiernaIzquierda} \mid \cdots
\end{array}
$$

Formalismo de la lógica de primer orden

Matías abrió un restaurante

$$\exists (x) \exists (e) \text{Restaurante}(x) \wedge \text{Abrir}(e, \text{Matías}, x)$$

Bases para el análisis semántico

- Una forma para representar el conocimiento
  - Lógicas descriptivas, lógica de primer orden

- Una forma para, a partir del lenguaje natural, saber qué conocimiento representar
  - Técnicas de extracción del conocimiento → análisis semántico

Bases para la extracción del conocimiento

- Principio de composición (y cómo aplicarlo)

- Hipótesis rule-to-rule (y cómo aplicarla)

Principio de composición

El significado de una oración se construye a partir del significado de sus partes

$$¿\ sgdo(oracion) = \sum sgdo(palabras)?$$

Principio de composición

El significado de una oración se construye a partir del significado de sus partes

$$¿\ sgdo(oracion) = \sum sgdo(palabras)?$$

Ejemplos:

- Análisis de sentimiento con búsqueda de palabras relevantes
- Métodos basados en BoW

Principio de composición

El significado de una oración se construye a partir del significado de sus partes

$$\cancel{¿\ sgdo(oracion) = \sum sgdo(palabras)}\ ?$$

María ama a Pedro ≠ Pedro ama a María ≠ Pedro María ama a

$$sgdo(oracion) = \sum \{orden, relaciones, agrupacion\}$$

Principio de composición

$$sgdo(oracion) = \sum \{orden, relaciones, agrupacion\}$$

El significado de una oración se construye a partir del significado de sus partes

$$\exists e\ Amar(e) \wedge Amante(e, Pedro) \wedge Amada(e, Maria)$$

Principio de composición

$$sgdo(oracion) = \sum \{orden, relaciones, agrupacion\}$$

El significado de una oración se construye a partir del significado de sus partes

Pedro ama a María → $\exists e\ Amar(e) \wedge Amante(e, Pedro) \wedge Amada(e, Maria)$

María ama a Pedro → $\exists e\ Amar(e) \wedge Amante(e, María) \wedge Amado(e, Pedro)$

Pedro María ama a → $Sin\ significado\ válido$

Hipótesis rule-to-rule

- Cada regla aplicada para constituir el análisis sintáctico tiene un significado equivalente en el análisis semántico

Gramáticas Libres de Contexto anotadas

a) NP → Matías $\{\lambda m.m(Matías)\}$

b) N → restaurante $\{\lambda r.Restaurante(r)\}$

c) Det → un $\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$

d) V_t → abrió $\{\lambda w.\lambda z.w(\lambda x.\exists e\ Abierto(e) \wedge PersonaQueAbre(e,z) \wedge CosaAbierta(e,x)\}$

Otra forma de representarlo:
- Abrir(e, z, x)

1) O → SN SV $\{SN.sem\ (SV.sem)\}$

2) SN → NP $\{NP.sem\}$

3) SN → Det N $\{Det.sem\ (N.sem)\}$

4) SV → V_t SN $\{Vt.sem\ (SN.sem)\}$

Se construyen con la hipótesis rule-to-rule

Rule-to-rule en anotaciones semánticas

- Asociar expresiones del cálculo lambda complejas, similares a una función, a las reglas léxicas.
- Copiar el valor semántico del único constituyente a la construcción cuando la regla gramatical solo tiene un constituyente.
- Aplicar la semántica de uno de los constituyentes de una regla gramatical a la semántica de otro de los constituyentes de la regla como si fuera una función.

- Foreach regla en GLC:
  - If (regla tiene símbolo termina a la derecha)
    - Lo rellenas directamente
  - Else if (regla tiene un único símbolo no terminal a la derecha)
    - Copias el símbolo terminal que va debajo
  - Else if (regla tiene dos o más símbolos no terminales a la derecha)
    - Haces una relación de subordinación

Rule-to-rule en anotaciones semánticas

Asociar expresiones del cálculo lambda complejas, similares a una función, a las reglas léxicas.

- NP → Matías $\{ \lambda m.m\ (Matías) \}$
- N → restaurante $\{ \lambda r.Restaurante(r) \}$
- Det → un $\{ \lambda P.\ \lambda Q.\exists x\ P(x) \wedge Q(x) \}$
- V_t → abrió $\{ \lambda w.\ \lambda z.w\ (\lambda x.\ \exists e\ Abierto(e) \wedge PersonaQueAbre(e,z) \wedge CosaAbierta(e,x)) \}$
- O → SN SV
- SN → NP
- SN → Det N
- SV → V_t SN

Rule-to-rule en anotaciones semánticas

Copiar el valor semántico del único constituyente a la construcción cuando la regla gramatical solo tiene un constituyente.

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV
- SN → NP { *NP.sem*}
- SN → Det N
- SV → V_t SN

Rule-to-rule en anotaciones semánticas

- Aplicar la semántica de uno de los constituyentes de una regla gramatical a la semántica de otro de los constituyentes de la regla como si fuera una función

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { *SN.sem(SV.sem)* }
- SN → NP { *NP.sem* }
- SN → Det N { *Det.sem(N.sem)* }
- SV → V_t SN { *V_t.sem(SN.sem)* }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Matías abrió un restaurante

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

{ λw. λz.w (λx. ∃e Abierto(e) ^PersonaQueAbre(e,z) ^ CosaAbierta(e,x)) }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { *SN.sem(SV.sem)* }
- SN → NP { *NP.sem* }
- SN → Det N { *Det.sem(N.sem)* }
- SV → V_t SN { *V_t.sem(SN.sem)* }

{ *Det.sem (N.sem)* }

{ *λP. λQ.∃x P(x) ^ Q(x) (λr.Restaurante(r))* }

{ *λQ.∃x λr.Restaurante(r)(x) ^ Q(x)* }

{ λQ.∃x Restaurante(x) ^ Q(x) }

Análisis Semántico en paralelo con el Sintáctico

1) Aplicamos la semántica de Det como función sobre la semántica de N:

$$Det.sem(N.sem) \Rightarrow \lambda P.\ \lambda Q.\exists x\ P(x) \wedge Q(x)\ (\lambda r.Restaurante(r))$$

2) Se resuelve instanciando la primera variable declareda en la función lambda:

$$\cancel{\lambda P}.\ \lambda Q.\exists x\ P(x) \wedge Q(x)\ (\lambda r.Restaurante(r)\ ) \Rightarrow \lambda Q.\exists x\ \lambda r.Restaurante(r)\ (x) \wedge Q(x)$$

3) Para terminar el cálculo, se resuelve lo que queda por instanciar de manera recursive:

$$\lambda Q.\exists x\ \cancel{\lambda r}.Restaurante(r)\ (x) \wedge Q(x) \Rightarrow \lambda Q.\exists x\ Restaurante(x) \wedge Q(x)$$

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { λm.m (Matías) }
- N → restaurante { λr.Restaurante(r) }
- Det → un { λP. λQ.∃x P(x) ^ Q(x) }
- Vt → abrió { λw. λz.w (λx. ∃e Abierto(e)
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { SN.sem(SV.sem) }
- SN → NP { NP.sem }
- SN → Det N { Det.sem(N.sem) }
- SV → V_t SN { V_t.sem(SN.sem) }

{ V_t.sem(SN.sem) }

{λw. λz.w (λx. ∃e Abierto(e)
^PersonaQueAbre(e,z)^CosaAbierta(e,x)) (λQ.∃x Restaurante((x) ^ Q(x))}

{λz. ∃x Restaurante(x) ^ λx. ∃e Abierto(e)
^PersonaQueAbre(e,z)^CosaAbierta(e,x) (x) }

{ λz. ∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e,z) ^ CosaAbierta(e,x) }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { *SN.sem(SV.sem)* }
- SN → NP { *NP.sem* }
- SN → Det N { *Det.sem(N.sem)* }
- SV → V_t SN { *V_t.sem(SN.sem)* }

{ *SN.sem(SV.sem)* }

{ *λm.m (Matías) (*
λz. ∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e,z) ^ CosaAbierta(e,x)) }

{*λz. ∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e,z) ^ CosaAbierta(e,x) (Matías)*}

{∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e, Matías) ^ CosaAbierta(e,x)}

Análisis Semántico en paralelo con el Sintáctico

- Ventajas
  - Descarta ramas inválidas semánticamente hablando
  - Dos análisis en un único cálculo
- Inconvenientes
  - Ineficiente computacionalmente
  - Complejo de construir

En resumen

- El análisis semántico busca formalizar el conocimiento
- Los modelos basados en tripletas
  - son más sencillos
  - permiten la aplicación de reglas lógicas.
- Los modelos basados en lógicas de primer orden
  - Son más complejos
  - Permiten paralelizar análisis sintáctico y semántico
  - Permiten una representación completa de la oración

En la próxima semana

¿Cómo aplicar técnicas de desambiguación del significado?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 8 – Semántica Léxica

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo aplicar técnicas de desambiguación del significado?

En el día de hoy

- Semántica Léxica:
  - **Significado de las palabras**
  - **Relaciones formales entre sentidos de las palabras**
  - **Recursos para la catalogación de significados, WordNet**
  - Relaciones informales entre sentidos de las palabras (similitud)
  - Desambiguación del sentido de las palabras

Significado de las palabras

- Semántica Léxica:
  - Campo de la lingüística que estudia el significado de las palabras y las relaciones de sentido que establecen entre ellas
  - Funciones:
    - Caracterizar el significado
      - Todos podemos dar una definición, pero caracterizar no es trivial
    - Estudiar las relaciones entre palabras
    - Estudiar la variación contextual del significado
      - https://www.hermanotemblon.com/laas-1000-acepciones-de-la-palabra-cojones/

Relaciones formales entre palabras

- Homonimia: homografía, homofonía
- Polisemia

- Sinónimos/antónimos
- Hipónimos/hiperónimos
- Merónimos/holónimos
- Metonimia

Relaciones formales entre palabras

- Homonimia: homografía, homofonía

homónimo, ma

Del lat. *homonȳmus*, y este del gr. ὁμώνυμος *homónymos*.

1. adj. Dicho de una persona o de una cosa: Que, con respecto de otra, tiene el mismo nombre. U. t. c. s.

2. adj. *Ling.* Dicho de una palabra: Que se pronuncia como otra, pero tiene diferente origen o significado muy distante; p. ej., *aya* 'niñera' y *haya* 'árbol'. U. t. c. s. m.

homógrafo, fa

De *homo-*[1] y *-grafo*.

1. adj. *Ling.* Dicho de una palabra: Que tiene la misma grafía que otra. Vino *'bebida' y* vino *'forma del verbo* venir*' son términos homógrafos.* U. t. c. s. m.

Requiere desambiguación

homófono, na

Del gr. ὁμόφωνος *homóphōnos*, de ὁμός *homós* 'igual' y φωνή *phōnḗ* 'sonido'.

1. adj. *Ling.* Dicho de una palabra: Que suena igual que otra, pero que tiene distinto significado y puede tener distinta grafía. U. t. c. s. m. *Aunque se distinguen ortográficamente,* tubo *y* tuvo *son homófonos.*

Genera dificultades al reconocer el lenguaje hablado

Relaciones formales entre palabras

- Polisemia
  - Una palabra es polisémica cuando tiene varios significados
    - Gato
    - Banco
    - Copa
  - La polisemia se aplica a los lemas (p.ej. marcha vs marcha)

Requiere desambiguación

Relaciones formales entre palabras

gato[1], ta SIN. / ANT.

Del lat. tardío *cattus*.

1. m. y f. Mamífero de la familia de los félidos, digitígrado, doméstico, de unos 50 cm de largo desde la cabeza hasta el arranque de la cola, que por sí sola mide unos 20 cm, de cabeza redonda, lengua muy áspera, patas cortas y generalmente pelaje suave y espeso, de color blanco, gris, pardo, rojizo o negro, empleado en algunos lugares para cazar ratones. U. en m. ref. a la especie.

SIN.: minino, michino, cucho[2], micho, morrongo, morroño, mozo[1].

2. m. y f. coloq. Persona nacida en Madrid.

SIN.: madrileño.

3. m. y f. *C. Rica y Nic.* Persona que tiene los ojos verdes o azules. U. t. c. adj.

4. m. y f. despect. coloq. *El Salv. y Méx.* **servidor** (|| persona que sirve como criado).

SIN.: criado, sirviente, servidor, chino[3], guachimán.

5. m. Máquina que sirve para levantar grandes pesos a poca altura, y que funciona con un engranaje y un trinquete de seguridad, o con una tuerca y un husillo.

SIN.: elevador, palanca, cric.

banco + SIN. / ANT.

Del fr. ant. *bank*, y este del germ. **banki*.

1. m. Asiento, con respaldo o sin él, en que pueden sentarse dos o más personas.

SIN.: banca, asiento, banqueta, taburete, grada[1], silla, poyo.

2. m. Madero grueso escuadrado que se coloca horizontalmente sobre cuatro pies y sirve de mesa para labores de carpinteros y otros artesanos.

3. m. En los mares, ríos y lagos navegables, bajo que se prolonga en una gran extensión.

SIN.: bajo, bajío, alfaque.

Polisémicas

Relaciones formales entre palabras

pila[1] + SIN. / ANT.

Del lat. *pila* 'columna'.

1. f. Conjunto de cosas puestas una sobre otra. *Pila de lana, de ladrillos.*

SIN.: montón, hacina, rimero, cúmulo.

2. f. coloq. **montón** (|| gran cantidad). *Pila DE años, DE gente.*

SIN.: montón, pilón[1], montonal, demonial, catizumbazo, catizumbada, bola, bolón, ponchada[2], ⚠ chinguero, macanazo, ⚠ vergazal, ⚠ chingo, mojón[1].

3. f. Conjunto de toda la lana que se corta cada año, perteneciente a un dueño.

pila[2] + SIN. / ANT.

Del lat. *pila* 'mortero'.

1. f. Pieza grande de piedra o de otra materia, cóncava y profunda, donde cae o se echa el agua para varios usos.

SIN.: fregadero, pilón[2], pileta, abrevadero.

2. f. Pieza de piedra, cóncava, con su pedestal de la misma materia, y tapa de madera, que hay en las iglesias parroquiales para administrar el sacramento del bautismo.

SIN.: baptisterio, fuente.

3. f. Parroquia o feligresía.

Homónimas / Homógrafas

Relaciones formales entre palabras

- Sinónimos

sinónimo, ma

Del lat. *synonȳmus*, y este del gr. συνώνυμος *synónymos*, de συν- *syn-* 'con-' y ὄνομα *ónoma* 'nombre'.

1. adj. *Ling.* Dicho de una palabra o de una expresión: Que, respecto de otra, tiene el mismo significado o muy parecido, como *empezar* y *comenzar*. U. t. c. s. m.

- Antónimos

antónimo, ma

De *anti-* y *-ónimo*.

1. adj. *Ling.* Dicho de una palabra: Que, respecto de otra, expresa una idea opuesta o contraria, como *virtud* y *vicio*, *claro* y *oscuro* o *antes* y *después*. U. t. c. s. m.

Son relaciones binarias

Relaciones formales entre palabras

- Hipónimos/hiperónimos

- A es más general que B
  - Animal es más general que perro
  - Perro es más general que dálmata

- Se dice que:
  - B es hipónimo de A
    - Perro es hipónimo de animal
  - A es hiperónimo de B
    - Perro es hiperónimo de dálmata

Relaciones formales entre palabras

- Merónimos/holónimos

- A es parte de B
  - La nariz forma parte de la cabeza
  - La fosa nasal forma parte de la nariz

- Se dice que:
  - A es merónimo de B
    - Nariz es merónimo de cabeza
  - B es holónimo de A
    - Nariz es holónimo de fosa nasal

Relaciones formales entre palabras

- Metonimia
  - Uso de una palabra en lugar de otra, valiéndose de una relación existente
- Causa por su efecto
- Contenedor por el contenido
- Símbolo por la cosa simbolizada
- Lugar por lo que en él se produce
- Marca comercial por objeto de la marca
- Obra por el nombre de su autor
- La parte por el todo
- El todo por la parte
- (…)

Genera sinonimia no catalogada

Recursos para la catalogación de significados

- Diccionarios
  - Recogen los significados de un lema

- Tesauros
  - Recogen las relaciones entre significados de un lema

- https://wordnet.princeton.edu/
- https://www.nltk.org/howto/wordnet.html

Relaciones informales entre palabras

- Similitud
  - Medida de relación entre palabras
    - Relación más relajada que la sinonimia
    - Dos palabras son más similares cuantas más características de su significado comparten, sin necesidad de llegar a ser sinónimos (casi sinónimos)
  - Diferente de la relación entre palabras
    - Balón vs esférico
    - Fútbol vs jugar
    - Fútbol vs baloncesto
    - Fútbol vs waterpolo
  - Es un criterio no binario

Relaciones informales entre palabras

- Métodos de cálculo de similitud
  - Medidas estadísticas
  - Basadas en tesauro
  - Basados en vectores

¿Para qué desambiguar?

- Una vez que hemos desambiguado:
  - Podemos generar un **resumen sintético**
  - Podemos generar una **traducción**
  - Podemos identificar la **respuesta a una pregunta**, bien con **razonamiento lógico** o bien con ***Information Retrieval***
  - Podemos afinar más en la **similitud entre oraciones**

En resumen

- La semántica léxica estudia los significados de las palabras
- La semántica léxica trabaja con los lemas
- Los significados se recogen en Diccionarios
- Las relaciones entre palabras se recogen en Tesauros

En la próxima semana

¿Cómo calcular la similitud de dos palabras?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 8 – Semántica Léxica

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo aplicar técnicas de desambiguación del significado?

En el día de hoy

- Semántica Léxica:
  - Significado de las palabras
  - Relaciones formales entre sentidos de las palabras
  - Recursos para la catalogación de significados, WordNet
  - Relaciones informales entre sentidos de las palabras (similitud)
  - Desambiguación del sentido de las palabras

Significado de las palabras

- Semántica Léxica:
  - Campo de la lingüística que estudia el significado de las palabras y las relaciones de sentido que establecen entre ellas
  - Funciones:
    - Caracterizar el significado
      - Todos podemos dar una definición, pero caracterizar no es trivial
    - Estudiar las relaciones entre palabras
    - Estudiar la variación contextual del significado
      - https://www.hermanotemblon.com/laas-1000-acepciones-de-la-palabra-cojones/

Relaciones formales entre palabras

- Homonimia: homografía, homofonía
- Polisemia

- Sinónimos/antónimos
- Hipónimos/hiperónimos
- Merónimos/holónimos
- Metonimia

Relaciones formales entre palabras

- Homonimia: homografía, homofonía

homónimo, ma

Del lat. *homonȳmus*, y este del gr. ὁμώνυμος *homónymos*.

1. adj. Dicho de una persona o de una cosa: Que, con respecto de otra, tiene el mismo nombre. U. t. c. s.

2. adj. *Ling.* Dicho de una palabra: Que se pronuncia como otra, pero tiene diferente origen o significado muy distante; p. ej., *aya* 'niñera' y *haya* 'árbol'. U. t. c. s. m.

homógrafo, fa

De *homo-*[1] y *-grafo*.

1. adj. *Ling.* Dicho de una palabra: Que tiene la misma grafía que otra. Vino *'bebida' y* vino *'forma del verbo* venir*' son términos homógrafos.* U. t. c. s. m.

Requiere desambiguación

homófono, na

Del gr. ὁμόφωνος *homóphōnos*, de ὁμός *homós* 'igual' y φωνή *phōnḗ* 'sonido'.

1. adj. *Ling.* Dicho de una palabra: Que suena igual que otra, pero que tiene distinto significado y puede tener distinta grafía. U. t. c. s. m. *Aunque se distinguen ortográficamente,* tubo *y* tuvo *son homófonos.*

Genera dificultades al reconocer el lenguaje hablado

Relaciones formales entre palabras

- Polisemia
  - Una palabra es polisémica cuando tiene varios significados
    - Gato
    - Banco
    - Copa
  - La polisemia se aplica a los lemas (p.ej. marcha vs marcha)

Requiere desambiguación

Relaciones formales entre palabras

gato[1], ta SIN. / ANT.

Del lat. tardío *cattus*.

1. m. y f. Mamífero de la familia de los félidos, digitígrado, doméstico, de unos 50 cm de largo desde la cabeza hasta el arranque de la cola, que por sí sola mide unos 20 cm, de cabeza redonda, lengua muy áspera, patas cortas y generalmente pelaje suave y espeso, de color blanco, gris, pardo, rojizo o negro, empleado en algunos lugares para cazar ratones. U. en m. ref. a la especie.

SIN.: minino, michino, cucho[2], micho, morrongo, morroño, mozo[1].

2. m. y f. coloq. Persona nacida en Madrid.

SIN.: madrileño.

3. m. y f. *C. Rica y Nic.* Persona que tiene los ojos verdes o azules. U. t. c. adj.

4. m. y f. despect. coloq. *El Salv. y Méx.* **servidor** (|| persona que sirve como criado).

SIN.: criado, sirviente, servidor, chino[3], guachimán.

5. m. Máquina que sirve para levantar grandes pesos a poca altura, y que funciona con un engranaje y un trinquete de seguridad, o con una tuerca y un husillo.

SIN.: elevador, palanca, cric.

banco + SIN. / ANT.

Del fr. ant. *bank*, y este del germ. **banki*.

1. m. Asiento, con respaldo o sin él, en que pueden sentarse dos o más personas.

SIN.: banca, asiento, banqueta, taburete, grada[1], silla, poyo.

2. m. Madero grueso escuadrado que se coloca horizontalmente sobre cuatro pies y sirve de mesa para labores de carpinteros y otros artesanos.

3. m. En los mares, ríos y lagos navegables, bajo que se prolonga en una gran extensión.

SIN.: bajo, bajío, alfaque.

Polisémicas

Relaciones formales entre palabras

pila[1] + SIN. / ANT.

Del lat. *pila* 'columna'.

1. f. Conjunto de cosas puestas una sobre otra. *Pila de lana, de ladrillos.*

SIN.: montón, hacina, rimero, cúmulo.

2. f. coloq. **montón** (|| gran cantidad). *Pila DE años, DE gente.*

SIN.: montón, pilón[1], montonal, demonial, catizumbazo, catizumbada, bola, bolón, ponchada[2], ⚠ chinguero, macanazo, ⚠ vergazal, ⚠ chingo, mojón[1].

3. f. Conjunto de toda la lana que se corta cada año, perteneciente a un dueño.

pila[2] + SIN. / ANT.

Del lat. *pila* 'mortero'.

1. f. Pieza grande de piedra o de otra materia, cóncava y profunda, donde cae o se echa el agua para varios usos.

SIN.: fregadero, pilón[2], pileta, abrevadero.

2. f. Pieza de piedra, cóncava, con su pedestal de la misma materia, y tapa de madera, que hay en las iglesias parroquiales para administrar el sacramento del bautismo.

SIN.: baptisterio, fuente.

3. f. Parroquia o feligresía.

Homónimas / Homógrafas

Relaciones formales entre palabras

- Sinónimos

sinónimo, ma

Del lat. *synonȳmus*, y este del gr. συνώνυμος *synónymos*, de συν- *syn-* 'con-' y ὄνομα *ónoma* 'nombre'.

1. adj. *Ling.* Dicho de una palabra o de una expresión: Que, respecto de otra, tiene el mismo significado o muy parecido, como *empezar* y *comenzar*. U. t. c. s. m.

- Antónimos

antónimo, ma

De *anti-* y *-ónimo*.

1. adj. *Ling.* Dicho de una palabra: Que, respecto de otra, expresa una idea opuesta o contraria, como *virtud* y *vicio*, *claro* y *oscuro* o *antes* y *después*. U. t. c. s. m.

Son relaciones binarias

Relaciones formales entre palabras

- Hipónimos/hiperónimos

- A es más general que B
  - Animal es más general que perro
  - Perro es más general que dálmata

- Se dice que:
  - B es hipónimo de A
    - Perro es hipónimo de animal
  - A es hiperónimo de B
    - Perro es hiperónimo de dálmata

Relaciones formales entre palabras

- Merónimos/holónimos

- A es parte de B
  - La nariz forma parte de la cabeza
  - La fosa nasal forma parte de la nariz

- Se dice que:
  - A es merónimo de B
    - Nariz es merónimo de cabeza
  - B es holónimo de A
    - Nariz es holónimo de fosa nasal

Relaciones formales entre palabras

- Metonimia
  - Uso de una palabra en lugar de otra, valiéndose de una relación existente
- Causa por su efecto
- Contenedor por el contenido
- Símbolo por la cosa simbolizada
- Lugar por lo que en él se produce
- Marca comercial por objeto de la marca
- Obra por el nombre de su autor
- La parte por el todo
- El todo por la parte
- (…)

Genera sinonimia no catalogada

Recursos para la catalogación de significados

- Diccionarios
  - Recogen los significados de un lema

- Tesauros
  - Recogen las relaciones entre significados de un lema

- https://wordnet.princeton.edu/
- https://www.nltk.org/howto/wordnet.html

Relaciones informales entre palabras

- Similitud
  - Medida de relación entre palabras
    - Relación más relajada que la sinonimia
    - Dos palabras son más similares cuantas más características de su significado comparten, sin necesidad de llegar a ser sinónimos (casi sinónimos)
  - Diferente de la relación entre palabras
    - Balón vs esférico
    - Fútbol vs jugar
    - Fútbol vs baloncesto
    - Fútbol vs waterpolo
  - Es un criterio no binario

Desambiguación del sentido de las palabras

¿Para qué desambiguar?

- Una vez que hemos desambiguado:
  - Podemos generar un **resumen sintético**
  - Podemos generar una **traducción**
  - Podemos identificar la **respuesta a una pregunta**, bien con **razonamiento lógico** o bien con ***Information Retrieval***
  - Podemos afinar más en la **similitud entre oraciones**

Antes de seguir, el concepto de “colocación”

- Dada una palabra cualquiera, en un texto cualquiera, la colocación de dicha palabra es la secuencia de N palabras previas, y N palabras posteriores, a la palabra que genera la colocación.

- Es decir, un N-grama generado por las palabras anteriores y posteriores

Métodos basados en conocimiento

- Lesk

Leemos el significado en el diccionario

- Lesk simplificado
  - Se lematiza cada uno de los sentidos de una palabra
  - Se lematiza la colocación
  - Se comparan ambas lematizaciones
- Lesk original
  - Se lematiza cada uno de los sentidos de una palabra
  - Se lematiza cada uno de los sentidos de las palabras colocadas
  - Se comparan las lematizaciones

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:
Empresa dedicada a realizar operaciones financieras con el dinero procedente de sus accionistas y de los depósitos de sus clientes.
(empresa, dedicar, realizar, operación,
financiera, dinero, accionista, depósito, cliente)

banco[2]:
Asiento, con respaldo o sin él, en que pueden sentarse dos o más personas.
(asiento, respaldo, poder, sentarse, persona)

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:

(empresa, dedicar, realizar, operación,
financiera, dinero, accionista, depósito, cliente)

banco[2]:

(asiento, respaldo, poder, sentarse, persona)

Contexto:

(prestar, dinero)

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:

(empresa, dedicar, realizar, operación, financiera, dinero, accionista, depósito, cliente)

banco[2]:

(asiento, respaldo, poder, sentarse, persona)

Contexto:

(prestar, dinero)

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:

(empresa, dedicar, realizar, operación, financiera, dinero, accionista, depósito, cliente)

1 solapamiento

banco[2]:

(asiento, respaldo, poder, sentarse, persona)

0 solapamientos

Contexto:

(prestar, dinero)

Lesk original

- Compara la firma de la palabra ambigua (signature), que se corresponde con su **definición en el diccionario**, con **las firmas de las palabras** de contexto (context), es decir, con las palabras vecinas a la palabra ambigua

*Me senté en un **banco** del parque*

banco1:
(empresa, dedicar, realizar, operación, financiera, dinero, accionista, depósito, cliente) — 0 solapamiento

banco2:
(asiento, respaldo, poder, sentarse, persona) — 2 solapamientos

Contexto:
sentar: poner a una persona en un asiento apoyado sobre sus nalgas
(poner, persona, asiento, apoyar, nalgas)
parque: Espacio que se dedica a jardines
(espacio,dedicar, jardin)

Lesk original

- Matriz de coincidencias y varios sentidos posibles en el contexto

parque1: Espacio que se dedica a jardines
(espacio, dedicar, jardin)

parque2: Estanque para cría de mariscos
(estanque, criar, marisco)

	banco1	banco2
sentar1	2	0
parque1	0	0
parque2	0	0

Sólo se elige un sentido

Métodos basados en conocimiento

- Ventajas
  - Muy explicables
  - Basta con disponer de un recurso básico (diccionario)

- Inconvenientes
  - No trabaja bien con entradas de diccionario muy cortas
    - Ampliar con sinónimos

  - Todas las palabras tienen el mismo peso
    - Aplicar un peso por colocación, POS Tag.

Aprendizaje no supervisado: Schütze

- Clusterización
  - Se calcula un vector de contexto *c* para cada aparición w_i de la palabra *w* en el corpus

    (con un bag of words de la colocación)

  - Se aplica un algoritmo de agrupamiento sobre los vectores de contexto c en un número predefinido de grupos, que definirán cada uno de los sentidos de la palabra w

    (con k-means)

  - Se calcula el centroide de cada grupo. Cada vector de un centroide será el vector representación de un sentido de la palabra w

    (promedio de cada dimensión)

Vectorizar texto: Empezamos con one-hot-encoding

Cada categoría se representa por un vector único

Color
Red
Green
Blue
Green

Red	Green	Blue
1	0	0
0	1	0
0	0	1
0	1	0

Y cada palabra es una categoría

Vectorizar texto: Empezamos con one-hot-encoding

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Hay 27 palabras diferentes → Se representan con vectores de dimensión 27

en	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lugar	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
la	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mancha	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cuyo	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nombre	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
no	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
quiero	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
acordarme	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ha	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mucho	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tiempo	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
que	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
vivía	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
hidalgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
los	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
lanza	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
adarga	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
antigua	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
rocín	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
flaco	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
galgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
corredor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Vectorizar texto: Bag of Words

- Recuento de palabras
  - O1: “antiguo, de un astillero antiguo”
  - O2: “antiguo cuadro”

	antiguo	astillero	cuadro	de	un
O1	2	1	0	1	1
O2	1	0	1	0	0

BoW TF

Clusterizar vectores

Schütze

- Ventajas
  - No hace falta un corpus etiquetado
  - Conceptualmente sencillo

- Inconvenientes
  - Mucho preprocesamiento para cada palabra
  - Baja explicabilidad, aunque explicable en última instancia

    (podríamos asignar palabras clave, con peso, a cada corpus)

Aprendizaje automático semisupervisado (bootstraping)

- Yarowsky

El corpus que tenemos es pequeño, y nos tenemos que apañar

- Entreno el clasificador con los (pocos) ejemplos que tengo
- Clasifico el resto de las instancias
- Las instancias más sólidamente clasificadas, las incorporo al corpus etiquetado
- Repito hasta que el error sea bajo, o me quede sin ejemplos

Similitud entre palabras

Similitud mediante tesauro

- Dos palabras son más similares si están cerca en el árbol del tesauro. Es decir, si hay pocos saltos entre ellas

- En la práctica, se utilizan relaciones de hiponimia e hiperonimia

- Entre sentidos: $$sim_{path(c_1,c_2)} = \frac{1}{pathlen(c_1, c_2)}$$

- Entre palabras: $$wordsim(w_1, w_2) = \max_{\substack{c_1 \in senses(w_1) \\ c_2 \in senses(w_2)}} sim(c_1, c_2)$$

Similitud mediante tesauro: ejemplo

- Similitud entre silla y mesa (chair and table)

- S: (n) **table**, tabular array (a set of data arranged in rows and columns) *"see table 1"*
- S: (n) **table** (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs) *"it was a sturdy table"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *has instance*
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*

- S: (n) **chair** (a seat for one person, with a support for the back) *"he put his coat over the back of the chair and sat down"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) seat (furniture that is designed for sitting on) *"there were not enough seats for all the guests"*
      - *direct hyponym* / *full hyponym*
      - *part meronym*
      - *member holonym*
      - *has instance*
      - ***direct hypernym*** / *inherited hypernym* / *sister term*
        - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*

path(silla, mesa) = 3+1 = 4
sim_path(silla, mesa) = 1/4 = 0.25

Similitud mediante tesauro: ejemplo

```python
from nltk.corpus import wordnet as wn
```

```python
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```python
# Referencias a los significados de cada palabra
orange_synsets = wn.synsets('naranja', lang='spa')
lemon_synsets = wn.synsets('limón', lang='spa')
print(orange_synsets)
print(lemon_synsets)
```

```
[Synset('orange.n.02'), Synset('orange.n.01'), Synset('orange.n.04')]
[Synset('lemon.n.04'), Synset('lemon.n.01')]
```

```python
# Recorriendo los synsets de cada palabra
for x in wn.synsets('manzana', lang='spa'):
  print("*"*15)
  print(x)
  print("Lemas:", x.lemma_names('spa')) # lemas de la palabra
  #print("Hipónimos:",x.hyponyms()) # hipónimos de la palabra
  print("Hiperónimos:", x.hypernyms()) # hiperónimos de la palabra
  print("Definición:", x.definition()) # definiciones de la palabra
  print("*"*15)
  print()
```

```
***************
Synset('apple.n.01')
Lemas: ['manzana']
Hiperónimos: [Synset('edible_fruit.n.01'), Synset('pome.n.01')]
Definición: fruit with red or yellow or green skin and sweet to tart crisp whitish flesh
***************

***************
Synset('eating_apple.n.01')
Lemas: ['manzana']
Hiperónimos: [Synset('apple.n.01')]
Definición: an apple used primarily for eating raw without cooking
***************
```

path(lemon, apple) = 3+1 = 4
sim_path(lemon, apple) = 1/4 = 0.25

path(lemon, orange) = 2+1 = 3
sim_path(lemon, orange) = 1/3 = 0.33

```python
# Similitud entre dos palabras en base al tesauro
print(wn.synset('orange.n.01').path_similarity(wn.synset('apple.n.01')))
print(wn.synset('orange.n.01').path_similarity(wn.synset('lemon.n.01')))
```

```
0.25
0.3333333333333333
```

Similitud mediante tesauro: problemas

- Subjetividad inherente al tesauro

- Valores discretos

- Se asume igual valor para todos los saltos
  - Alternativas:
    - Normalizar las distancias según la profundidad del árbol
    - Asociar distancias diferentes
    - Métodos basados en la teoría de la información (Resnik)

Similitud mediante vectores

- Vectores extraídos como Word Embedings
  - Word2Vec, Glove, Fasttext, ELMO, BETO, BERT

Inciso: ¿qué es un embedding?

- Representación vectorial de una palabra en la que:
  - Los vectores guardan un tipo de relación con la información de la palabra.
  - Los vectores se calculan con técnicas no supervisadas

Male-Female | Verb tense | Country-Capital

Similitud mediante vectores

- Vectores extraídos como Word Embedings
  - Word2Vec, Glove, Fasttext, ELMO, BETO, BERT

- Métricas matemáticas bien conocidas
  - Distancia euclídea
  - Similitud de coseno
  - Pearson
  - Jaccard
  - Levensthein
  - 17 types of similarity and dissimilarity measures used in data science

Similitud mediante vectores

- Ventajas
  - Facilidad de cálculo
  - No hace falta tesauro o corpus etiquetado
  - Muy buenos resultados

- Desventajas
  - Falta de explicabilidad

En resumen (bis)

- La semántica léxica estudia los significados de las palabras
- La semántica léxica trabaja con los lemas
- Los significados se recogen en Diccionarios
- Las relaciones entre palabras se recogen en Tesauros
- Desambiguación como tarea relevante en semántica léxica
  - Métodos basados en conocimiento
  - Métodos basados en vectores
- La similitud es una relación más relajada
  - Métodos basados en tesauro
  - Métodos basados en vectores

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 5 – Modelado del Lenguaje

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

Dada una secuencia de texto, ¿cómo podemos predecir el texto siguiente?

Encuesta previa

- ¿Funciona bien el predictor de texto del teclado móvil?
- ¿Funciona bien el reconocimiento del habla del teclado del móvil?
- ¿Sabes qué es el análisis de sentimientos?
- ¿Sabéis lo que es un LLM?

En el día de hoy

- Introducción al Modelado del Lenguaje Estadístico
- Modelado de Lenguaje con N-Gramas
  - Cómo crearlos
  - Cómo medirlos
  - Técnicas de suavizado

- Introducción a los Modelos de Representación
- Modelos de Representación Vectorial
  - BoW
  - TF
  - TF-IDF

Introducción al modelado del lenguaje estadístico

- ¿Qué es?
  - "*Models that assign **probabilities to upcoming words**, or **sequences of words** language model in general, are called language models or LMs*" (Jurafsky, 2024)

- ¿Para qué sirve?
  - Teclados predictivos
  - Traducción automática
  - Speech to text
  - Generación de textos

- ¿Cómo se hace?
  - Estadístico

Introducción al modelado del lenguaje estadístico

Resumen. El texto habla de la historia de un estudiante….

~~Large~~ Language Model
(~~L~~LM)

Resume este texto. “Érase una vez un estudiante de Ingeniería Informática….”

Modelado de Lenguaje con N-Gramas

- Tokens (palabras)
  - [“En”, “un”, “lugar”, “de”, “La_Mancha”]

- **Bigramas**
  - [[“En”, “un”], [“un”,”lugar”],[“lugar”, “de”], [“de”, “La_Mancha”]]

- **Trigramas**
  - [[“En”, “un”, “lugar”], [“un”,”lugar”, “de”], [“lugar”, “de”, “La_Mancha”]]

- **N-gramas**
  - […]

Modelado de Lenguaje con N-Gramas

$$P(chinese \mid i\ want\ to\ eat) = \frac{C(i\ want\ to\ eat\ chinese)}{C(i\ want\ to\ eat)}$$

1. Consideración: Aplicamos la regla de la cadena

$$P(w_i{:}n) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \dots P(w_n|w_{1:n-1}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

2. Simplificación: Aplicamos la hipótesis de Markov

$$P(w_i{:}n) \sim \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Modelado de Lenguaje con N-Gramas

$$P(\text{eat} \mid i\ want\ to) = \frac{C(i\ want\ to\ eat)}{C(i\ want\ to)}$$

1. Consideración: Aplicamos la regla de la cadena

$$\boxed{\begin{aligned} &P\ (I\ want\ to\ eat) \\ &= P(I) \times P(want|I) \times P(to|I\ want) \times P(eat|I\ want\ to) \end{aligned}}$$

2. Simplificación: Aplicamos la hipótesis de Markov

$$\boxed{\begin{aligned} &P\ (I\ want\ to\ eat) \\ &= P(I) \times P(want|I) \times P(to|want) \times P(eat|to) \end{aligned}}$$

Modelado de Lenguaje con N-Gramas

- ¿Cuál es la probabilidad de que, dada una palabra, aparezca otra palabra concreta?
- **Para un texto de referencia dado**, ¿Cuál es la probabilidad de que, dada una palabra, aparezca otra palabra concreta?

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- ¿Cuál es la probabilidad de que tras la palabra “want” aparezca la palabra to?

$$P\ (to|want) = \frac{Apariciones\ de\ "want\ to"}{Apariciones\ de\ "want"}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

Ejemplo recortado, con lo que las probabilidades no van a sumar 1

- ¿Cuál es la probabilidad de que tras la palabra “want” palabra to?

$$P\,(to|want) = \frac{Apariciones\ de\ "want\ to"}{Apariciones\ de\ "want"}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- La probabilidad en un bigrama es fácil, ¿y en un trigrama?

$$P\ (to|I\ want) = \frac{Apariciones\ de\ "I\ want\ to"}{Apariciones\ de\ "I\ want"} = \frac{??}{827}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- Ya somos capaces de:
  - Dada una secuencia, calcular la probabilidad de una siguiente palabra dada
  - Calcular la probabilidad de una secuencia dada

Siempre, en un corpus de referencia

Modelado de Lenguaje con N-Gramas

- Problemas del modelo
  - Hay palabras que no aparecen en el corpus
  - Hay bigramas que no aparecen en el corpus (y todas sus secuencias serán cero)

Técnicas de suavizado

- ¿Por qué el suavizado de modelos?
- Técnica de Laplace y técnica add-k
- Backoff e interpolación

¿Por qué el suavizado de modelos?

Corpus de entrenamiento:

- *El gato come pescado*
- *El gato anda despacio*
- *El gato está durmiendo*
- *El perro salta mucho*

Corpus de inferencia/test:

- *El gato salta mucho*
- *El gato come carne*

$$P(salta|gato) = 0$$

$$P(El|Init) * P(El|gato) * P(salto|gato) * P(mucho|salta) = 0$$

Autocompletado: El gato salta… -> secuencia imposible de completar

Técnica de Laplace y técnica add-k

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

+1

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Técnica de Laplace y técnica add-k

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

+k

Backoff e interpolación

- Backoff
  - Se utilizan N gramas cada vez más pequeños, hasta encontrar uno que sea distinto de cero

$$P(eat|I\ want\ to) \approx P(to|I\ want) \approx P(want|I) \approx P(I)$$

- Interpolación
  - Se calcula como combinación lineal de n-gramas de menor orden

$$P(eat|I\ want\ to) \approx \lambda_1 P(to|I\ want) \times \lambda_2 P(want|I) \times \lambda_3 P(I)$$

Evaluación

- Nos dice en qué medida funciona un modelo

- Extrínseca
  - Se prueba en la aplicación final
    (p.ej. *Precission* y *recall* en la predicción de un teclado móvil)

- Intrínseca
  - Perplejidad

Perplejidad

- Cuanto menor sea el valor, mejor es el modelo

$$PP(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}}$$

Consideración: regla de la cadena

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1, \dots, w_{i-1})}}$$

Simplificación: hipótesis de Markov

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Introducción los modelos de representación vectorial

- Qué es
  - "***Vector semantics***, *which instantiates this linguistic vector semantics embeddings hypothesis by learning* ***representations*** *of the* ***meaning of words****, called embeddings, directly from their* ***distributions*** *in text*" (Jurafsky, 2024)

- Para qué sirve
  - Análisis de sentimientos
  - Topic detection
  - Clasificación de textos

- Cómo se hace?
  - Vectorial

Nuestro texto de referencia

En un lugar de La Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Empezamos con one-hot-encoding

Hay 27 palabras diferentes → Se representan con vectores de dimensión 27

en	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lugar	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
la	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mancha	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cuyo	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nombre	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
no	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
quiero	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
acordarme	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ha	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mucho	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tiempo	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
que	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
vivía	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
hidalgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
los	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
lanza	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
adarga	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
antigua	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
rocín	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
flaco	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
galgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
corredor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Características de one-hot-enconding

- Es fácil de generar y manejar
- Todos los vectores son ortogonales entre sí
- En redes neuronales, cada palabra activaría una neurona de entrada diferente
- Dimensión del vector potencialmente muy elevada
- El vector no contiene información sobre la palabra, es sólo una asignación arbitraria

Modelos de lenguaje con Vectores

- Bag of Words (BoW), es una técnica que genera un único vector que contiene todas las palabras de un texto.

- BoW binario
- BoW TF
- BoW WTF
- BoW TF-iDF

BoW binario

- 1 si la palabra aparece. 0 si la palabra no aparece.
  - “antiguo, de un astillero antiguo”

antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
BoW binario	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

BoW TF

- Recuento de palabras
  - “antiguo, de un astillero antiguo”

antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
BoW TF	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0

BoW WTF

- Frecuencia de aparición de las palabras
  - “antiguo, de un astillero antiguo”

antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
BoW WTF	0	0,2	0	0,2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,2	0	0,4	0	0	0	0	0

$$WTF(t_i, d_j) = \frac{f_{ij}}{\sum_{t_p \in d_j} f_{pj}}$$

BoW TF-IDF (representación global)

- Se pondera TF con respecto a las apariciones del mismo término en otros documentos
  - Nos dice si la palabra es un factor diferencial de este texto con respecto a otros

$$TFIDF(t_i, d_j) = f_{ij} \times \log\left(\frac{N}{df(t_i)}\right)$$

- Ejemplo (no estricto, pero muy ilustrativo)

  https://flowingdata.com/2018/09/18/cuisine-ingredients/

  - En el ejemplo,
    - “most used” sería TF
    - “cuisine-specific” sería TF-IDF

Una vez que tengo el texto como vector

Una vez que tengo el texto como vector

- Problemas
  - Vectores de entrada potencialmente muy grandes
  - Vectores de entrada que NO tienen relación con el significado
  - Modelo incapaz de entender la secuencialidad del texto

En resumen

- Modelado del lenguaje: estadísticos
  - Modelo que nos permite conocer la probabilidad de aparición de un N-Grama
  - Aplicaciones
    - Teclados móviles
    - Traducción
    - Text-to-speech

- Modelos de representación vectorial
  - Modelo que nos permite construir un vector que representa a un texto
  - Aplicaciones
    - Procesado neuronal
    - Análisis de sentimientos

Introducción a la Actividad 2

- 3 referencias bibliográficas (posteriores a 1995…)
- Soluciones comerciales
- Evolución histórica, transferencia investigación – empresa, retos

Etiquetado morfosintáctico (valor real: puntos)	Descripción	Puntuación máxima (puntos)	Peso %
Referencias adecuadas	Para cada referencia aportada, se trata de un trabajo experimental sobre el tema objeto de estudio en el que se pueden observar resultados empíricos.	2	20%
Referencias bien analizadas	El análisis de las referencias en cuanto a tecnologías utilizadas, metodología experimental y resultados obtenidos refleja adecuadamente el contenido de los artículos y está planteado de una forma clara y ordenada.	4	40%
Soluciones comerciales adecuadas	La solución comercial identificada es, efectivamente, un producto que utiliza PLN para resolver el tema objeto de estudio.	1	10%
Soluciones comerciales bien analizadas	El análisis de las soluciones comerciales en cuanto a las preguntas planteadas en el enunciado de la actividad es coherente, está bien argumentado y no presenta errores de redacción.	2	20%
Preguntas finales	Las respuestas otorgadas a las preguntas finales son coherentes, están bien argumentadas y no presenta errores de redacción.	1	10%
		10	**100 %**

- Traducción automática
- Autocompletado y generación automática de resúmenes
- Análisis de sentimientos
- Question Answering
- Reconocimiento automático del habla
- Text-to-speech

En la próxima semana

¿Cómo podemos representar una palabra en forma vectorial?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 6 – Modelado de Lenguaje Neuronal

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Encuesta previa

- ¿Qué es un Word embedding?

- ¿Qué técnica introduce la idea de partir las palabras en tokens menores?

En el día de hoy

- Embeddings (recordatorio)
- Uso de Embeddings para Modelado del Lenguaje y Clasificación
- Redes Neuronales Recurrentes y LSTM
- Otro tipo de Word Embeddings: ELMo

Word2Vec: Idea intuitiva

King – Man = Queen – Woman.

- Disponer de vectores **densos** (que no sean solo un 1 y muchos 0s) que representen el **significado de las palabras** dentro de un **espacio vectorial**
- La representación del significado se basa en la **perspectiva distribucional**
- Con estos vectores se pueden hacer operaciones como las del ejemplo

Embeddings (recordatorio)

$$\begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} \times \begin{pmatrix} 0.4 & \dots & 2.1 \\ \vdots & \ddots & \vdots \\ -1.3 & \dots & 1.1 \end{pmatrix} = \begin{pmatrix} 0.4 \\ \vdots \\ 2.1 \end{pmatrix}$$

Lookup table

Ejemplos de uso de embeddings

Modelado de lenguaje

(Jurafsky, 2021)

Ejemplos de uso de embeddings

Clasificación de texto

https://michael-fuchs-python.netlify.app/2021/09/01/nlp-word-embedding-with-gensim-for-text-classification/

Clasificación de textos con redes clásicas

Clasificación de textos con redes clásicas

Opciones de Lookup table:

- Preentrenada en otros datos
- Entrenada desde 0 dentro del propio modelo
- Preentrenada en otros datos y reentrenada en el propio modelo

Clasificación de textos con redes clásicas

Clasificación de textos con redes clásicas

Extra: ejemplos de cálculos con NN

https://www.linkedin.com/posts/tom-yeh_deeplearning-neuralnetworks-activity-7138659440077000705-OjXh

Redes recurrentes

Fang, Hao & Jiang, Man & Lu, Jingyan & Qin, Gong & Tao, & Tang, Peifu & Yunjun, Yan & Yun, Luo & Liu, Jingtian. (2020). A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages. Remote Sensing. 12. 256. 10.3390/rs12020256.

Tipos de RNN

https://stats.stackexchange.com/questions/129411/how-can-recurrent-neural-networks-be-used-for-sequence-classification

Para Clasificación de Textos

Para Modelado del Lenguaje

Usamos una aproximación Many to Many donde, a cada paso, se tiene la lista de probabilidades para la siguiente palabra

(Jurafsky, 2021)

Extra: ejemplos de cálculos con RNN

https://www.linkedin.com/posts/tom-yeh_deeplearning-neuralnetworks-generativeai-activity-7142506286788620288-jLN4

Redes recurrentes. Influencia decreciente

Redes LSTM

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

Redes LSTM

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

Redes LSTM

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

Redes LSTM bidireccionales

*He usado mi **gato** para elevar el coche.*

- Una RNN se puede usar para Traducción.
- Es importante tener la información previa y posterior para hacer traducciones (e.j., “car jack”, no “cat”)

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

Otro tipo de Word Embeddings

- Word2Vec, FastText o Glove
  - Dan un mismo vector de embeddings para cada palabra, con lo que no resuelven el problema de **palabras con varios significados**

- Existen otro tipo de representaciones, basadas en redes **LSTM bidireccionales,** que generan embeddings dependientes del contexto **(contextual representation)**
  - El embedding hace mejor uso del contexto
  - Un embedding diferente **para una misma palabra** en oraciones diferentes
  - oraciones diferentes

ELMo (Embeddings from Language Models)

https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

ELMo Demo

- Ejemplo extraído de https://www.geeksforgeeks.org/overview-of-word-embedding-using-embeddings-from-language-models-elmo/

https://colab.research.google.com/drive/1ivu0eVa6uacmjS_gq8P4SDGJR-8bbHhm?usp=sharing

En resumen

- Los embeddings generan **vectores que representan palabras**.
- Estos embeddings se pueden usar como **input** de otros modelos de NN (e.j., para clasificación de sentimientos).
- Las aplicaciones de NLP basadas en RNN tienen en cuenta el **aspecto secuencial** de las palabras.
- Existen otro tipo de representaciones de Word embeddings que tienen en cuenta el **contexto**, como es el caso de ELMo.

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Sesión de Refuerzo del Exámen

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Decíamos ayer…

- **Procesamiento del Lenguaje Natural:**
  Campo de la **Inteligencia Artificial** que tiene como objetivo principal hacer que las **máquinas** sean capaces de realizar tareas que involucren el lenguaje humano.

- **Morfología y Análisis Morfológico:**
  entender e identificar las partes con entidad propia que forman una palabra

- **Etiquetado Morfosintáctico:**
  identificar la funcionalidad de las palabras como parte de la oración (POS) usando modelos de lenguaje (HMM - Viterbi)

- **Gramáticas para el Análisis Sintáctico:**
  uso de gramáticas para formalizar la información sintagmática y para crear analizadores sintácticos

- **Análisis Sintáctico:**
  modelado de la información sintáctica mediante gramáticas (GLC y CNF) y uso de algoritmos basados en programación dinámica para trabajar con la ambigüedad estructural (CKY/CKY Prob.)

Decíamos ayer…

- **Semántica**
  Campo de la **Inteligencia Artificial** que tiene como objetivo principal hacer que las **máquinas** sean capaces de realizar tareas que involucren el lenguaje humano.

- **Análisis Semántico**
  modelar el conocimiento semántico de los textos extendiendo la información sintáctica mediante GLCs con anotaciones semánticas

- **Semántica Léxica**
  trabajar sobre el problema de la desambiguación del sentido de las palabras, definir relaciones y similitud entre sentidos de palabras

Examen Procesamiento del Lenguaje Natural

- Examen teórico práctico
- Abarca del Tema 1 al Tema 10
- 2 partes: teórica y práctica

Examen Procesamiento del Lenguaje Natural

- Parte teórica (4 puntos)
  - En general, una pregunta teórica de cada tema
  - Casa pregunta del tipo test tiene el mismo valor
  - Cada pregunta errónea restará un 0.25 del valor de la pregunta sobre el total (es decir, -0.1 del total del examen)
  - Preguntas tanto teóricas (similares a los tests de la asignatura) como de cálculos sencillos.

- Parte práctica (6 puntos)
  - Un ejercicio teórico práctico sobre cada parte del temario
    Etiquetado morfosintáctico (3 puntos)
    Análisis Sintáctico (3 puntos)
    Semántica Léxica (3 puntos)
    Modelos de representación vectorial (3 puntos)

Ejemplos de preguntas de test

En un corpus de 4500 documentos, la palabra "bondad" aparece en 50 documentos.
Si en un documento específico, la palabra "bondad" aparece 10 veces, ¿cuál es el valor del TF-IDF para "bondad" en ese documento?
(nota: usa el logaritmo en base 10)

N = 4500
df_i = 50
tf_ij = 10

$$tfidf = tf_{ij} \times \log_{10}\left(\frac{N}{df_i}\right) = 10 \times \log_{10}\left(\frac{4500}{50}\right) = 19.54$$

Ejemplos de preguntas de test

Dado un modelo de etiquetado POS, se observa que la etiqueta "NN" (sustantivo) aparece en el corpus 4500 veces y la etiqueta "VB" (verbo base) aparece 32000 veces. Si la palabra "vivir" aparece etiquetada como "NN" 50 veces y como "VB" 550 veces,

-> ¿cuál es la probabilidad de emisión de "vivir" como "NN"?
-> ¿cuál es la probabilidad de emisión de "vivir" como "VB"?

C (NN) = 4500
C (VB) = 32000

$$P(vivir|NN) = \frac{C(vivir, NN)}{C(NN)} = \frac{50}{4500} = 0.011$$

C (vivir, NN) = 50
C (vivir, VB) = 550

$$P(vivir|VB) = \frac{C(vivir, VB)}{C(VB)} = \frac{550}{32000} = 0.017$$

Examen Procesamiento del Lenguaje Natural

- Parte práctica (6 puntos)

ANÁLISIS MORFOSINTÁCTICO

ANÁLISIS SINTÁCTICO

SEMÁNTICA LÉXICA

MODELOS DE REPRESENTACIÓN VECTORIAL

Análisis morfosintáctico

- Se tiene como objetivo crear un etiquetador morfosintáctico. Para ello, se ha decidido crear un HMM Bigrama mediante el cual se va a asignar automáticamente a cada palabra del texto una categoría gramatical. Para ello, cuenta con un corpus de texto anotado manualmente de forma tal que cada palabra se ha anotado con una de estas etiquetas: NOUN, ADJ, VERB, ADV, DET, OTHER.

- A partir de este corpus anotado, se ha calculado la tabla de probabilidades de transición y la tabla de probabilidades de emisión (INIT denota el estado de inicio de secuencia y FINAL el de final de secuencia).

- A partir de ellas, se aplica el algoritmo de Viterbi para construir la matriz (se adjunta al final) y determinar la secuencia de categorías gramaticales más probable para la siguiente secuencia:
    - “La carga es pesada”.

- Una vez construida la matriz, se procede a aplicar el algoritmo de decodificación que le permite asignar a cada palabra de la secuencia su correspondiente categoría gramatical.

- Se pide que responda a las siguientes preguntas:

1. En el corpus se han contado 4.000.000 de palabras con etiqueta ADJ, y la palabra “pesada” aparece en el corpus 30.000 veces con dicha etiqueta. ¿Cuál es el valor en la tabla de emisión para la palabra “pesada” siendo esta ADJ? ¿Cómo se calculan los valores de la tabla de emisión y qué representan?

2. A partir de la matriz de Viterbi vista arriba, calcule el valor de las celdas que faltan y utilice el algoritmo de decodificación de Viterbi para determinar la categoría más probable para cada palabra de la secuencia

Análisis morfosintáctico

En el corpus se han contado **4.000.000 de palabras con etiqueta ADJ**, y **la palabra "pesada" aparece en el corpus 30.000 veces con dicha etiqueta.** ¿Cuál es el **valor en la tabla de emisión para la palabra "pesada" siendo esta ADJ**? ¿Cómo se calculan los valores de la tabla de emisión y qué representan?

	La	carga	es	pesada
NOUN	0.0	0.007	0.0	0.0
ADJ	0.0	0.0	0.0	¿?
VERB	0.0	0.005	0.01	0.0
ADV	0.0	0.0	0.0	0.0
DET	0.01	0.0	0.0	0.0
OTHER	0.0	0.0	0.0	0.0

Tabla de probabilidades de emisión

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$C(adj) = 4M$$

$$C(adj, pesada) = 30k$$

$$P(pesada|adj) = \frac{30k}{4M} = 0{,}0075$$

Análisis morfosintáctico

A partir de la matriz de Viterbi vista arriba, calcule el valor de las celdas que faltan y utilice el algoritmo de decodificación de Viterbi para determinar la categoría más probable para cada palabra de la secuencia

	NOUN	ADJ	VERB	ADV	DET	OTHER	FINAL
INIT	0.3	0.1	0.1	0.1	0.3	0.1	0.0
NOUN	0.1	0.3	0.2	0.1	0.1	0.1	0.1
ADJ	0.1	0.3	0.2	0.1	0.1	0.1	0.1
VERB	0.2	0.1	0.2	0.2	0.1	0.1	0.1
ADV	0.1	0.1	0.1	0.3	0.1	0.1	0.1
DET	0.5	0.3	0.0	0.0	0.2	0.1	0.0
OTHER	0.2	0.2	0.2	0.1	0.1	0.1	0.1

Tabla de probabilidades de transición

	La	carga	es	pesada
NOUN	0.0	0.007	0.0	0.0
ADJ	0.0	0.0	0.0	¿?
VERB	0.0	0.005	0.01	0.0
ADV	0.0	0.0	0.0	0.0
DET	0.01	0.0	0.0	0.0
OTHER	0.0	0.0	0.0	0.0

Tabla de probabilidades de emisión

INIT	1.0					
NOUN	0.0	0.0	1.05e-05	0.0	0.0	
ADJ	0.0	0.0	0.0	0.0	¿?	
VERB	0.0	0.0	0.0	2.1e-8	0.0	
ADV	0.0	0.0	0.0	0.0	0.0	
DET	0.0	0.003	0.0	0.0	0.0	
OTHER	0.0	0.0	0.0	0.0	0.0	
FINAL						¿?
		La	carga	es	pesada	

Matriz de Viterbi

Análisis morfosintáctico

A partir de la matriz de Viterbi vista arriba, calcule el valor de las celdas que faltan y utilice el algoritmo de decodificación de Viterbi para determinar la categoría más probable para cada palabra de la secuencia

INIT	1.0					
NOUN	0.0	0.0	1.05e-05	0.0	0.0	
ADJ	0.0	0.0	0.0	0.0	¿?	
VERB	0.0	0.0	0.0	2.1e-8	0.0	
ADV	0.0	0.0	0.0	0.0	0.0	
DET	0.0	0.003	0.0	0.0	0.0	
OTHER	0.0	0.0	0.0	0.0	0.0	
FINAL						¿?
		La	carga	es	pesada	

	NOUN	ADJ	VERB	ADV	DET	OTHER	FINAL
INIT	0.3	0.1	0.1	0.1	0.3	0.1	0.0
NOUN	0.1	0.3	0.2	0.1	0.1	0.1	0.1
ADJ	0.1	0.3	0.2	0.1	0.1	0.1	0.1
VERB	0.2	0.1	0.2	0.2	0.1	0.1	0.1
ADV	0.1	0.1	0.1	0.3	0.1	0.1	0.1
DET	0.5	0.3	0.0	0.0	0.2	0.1	0.0
OTHER	0.2	0.2	0.2	0.1	0.1	0.1	0.1

Tabla de probabilidades de transición

	La	carga	es	pesada
NOUN	0.0	0.007	0.0	0.0
ADJ	0.0	0.0	0.0	0,0075
VERB	0.0	0.005	0.01	0.0
ADV	0.0	0.0	0.0	0.0
DET	0.01	0.0	0.0	0.0
OTHER	0.0	0.0	0.0	0.0

Tabla de probabilidades de emisión

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t)$$

$$V[4] = v_3 \times P_{transicion}(adj|verb) \times P_{emision}(pesada|adj)$$

$$V[4] = 2.1e^{-8} \times 0.1 \times 0.0075 = 1.575e^{-11}$$

Análisis morfosintáctico

A partir de la matriz de Viterbi vista arriba, calcule el valor de las celdas que faltan y utilice el algoritmo de decodificación de Viterbi para determinar la categoría más probable para cada palabra de la secuencia

INIT	1.0					
NOUN	0.0	0.0	1.05e-05	0.0	0.0	
ADJ	0.0	0.0	0.0	0.0	$1.575e^{-11}$	
VERB	0.0	0.0	0.0	2.1e-8	0.0	
ADV	0.0	0.0	0.0	0.0	0.0	
DET	0.0	0.003	0.0	0.0	0.0	
OTHER	0.0	0.0	0.0	0.0	0.0	
FINAL						¿?
		La	carga	es	pesada	

Matriz de Viterbi

	NOUN	ADJ	VERB	ADV	DET	OTHER	FINAL
INIT	0.3	0.1	0.1	0.1	0.3	0.1	0.0
NOUN	0.1	0.3	0.2	0.1	0.1	0.1	0.1
ADJ	0.1	0.3	0.2	0.1	0.1	0.1	0.1
VERB	0.2	0.1	0.2	0.2	0.1	0.1	0.1
ADV	0.1	0.1	0.1	0.3	0.1	0.1	0.1
DET	0.5	0.3	0.0	0.0	0.2	0.1	0.0
OTHER	0.2	0.2	0.2	0.1	0.1	0.1	0.1

Tabla de probabilidades de transición

	La	carga	es	pesada
NOUN	0.0	0.007	0.0	0.0
ADJ	0.0	0.0	0.0	¿?
VERB	0.0	0.005	0.01	0.0
ADV	0.0	0.0	0.0	0.0
DET	0.01	0.0	0.0	0.0
OTHER	0.0	0.0	0.0	0.0

Tabla de probabilidades de emisión

$$V[final] = v_4 \times P_{transicion}(final|adj)$$
$$V[final] = 1.575e^{-11} \times 0.1 = 1.575e^{-12}$$

(La, det) (carga, noun) (es, verb) (pesada, adj)

Análisis sintáctico - CKY

Se parte de una gramática que dispone de reglas que cuentan con un valor de probabilidad (entre paréntesis) asociado, obtenidas de un treebank de 2M de secuencias etiquetadas en cuanto a análisis sintáctico.
A partir de esta gramática, con su información probabilística asociada, se hace uso del algoritmo CKY probabilístico para construir la tabla CKY y así construir el árbol de análisis sintáctico para la siguiente oración:
“Ella ve un planeta con un telescopio”.

T = {ella, ve, un, planeta, con, telescopio}
N = {O, SN, SV, D, N, P, Pron}
R = {
1. O → Pron SV (1.0)
2. SN → D N (1.0)
3. N → N SP (0.2)
4. SP → P SN (0.1)
5. SV→ V SN (0.7)
6. SV → SV SP (0.3)
a. Pron → ella (1.0)
b. V → ve (1.0)
c. D → un (1.0)
d. N → planeta (0.4)
e. N → telescopio (0.4)
f. P → con (1.0)
}
I = O

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	¿? [2,4]	[2,5]	[2,6]	¿? [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

Análisis Sintáctico - CKY

1. En la tabla CKY construida mediante el algoritmo CKY probabilístico para la secuencia “Ella ve un planeta con un telescopio”, falta por **calcular el valor de las celdas [2,4] y [2,7]**. ¿Qué **constituyentes** se han de colocar en ella y con qué valor de **probabilidad**?

T = {ella, ve, un, planeta, con, telescopio}
N = {O, SN, SV, D, N, P, Pron}
R = {
1. O → Pron SV (1.0)
2. SN → D N (1.0)
3. N → N SP (0.2)
4. SP → P SN (0.1)
5. SV→ V SN (0.7)
6. SV → SV SP (0.3)
a. Pron → ella (1.0)
b. V → ve (1.0)
c. D → un (1.0)
d. N → planeta (0.4)
e. N → telescopio (0.4)
f. P → con (1.0)
}
I = O

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	¿? [2,4]	[2,5]	[2,6]	¿? [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

Análisis Sintáctico - CKY

1. En la tabla CKY construida mediante el algoritmo CKY probabilístico para la secuencia “Ella ve un planeta con un telescopio”, falta por **calcular el valor de las celdas [2,4] y [2,7]**. ¿Qué **constituyentes** se han de colocar en ella y con qué valor de **probabilidad**?

T = {ella, ve, un, planeta, con, telescopio}
N = {O, SN, SV, D, N, P, Pron}
R = {
1. O → Pron SV (1.0)
2. SN → D N (1.0)
3. N → N SP (0.2)
4. SP → P SN (0.1)
5. SV→ V SN (0.7)
6. SV → SV SP (0.3)
a. Pron → ella (1.0)
b. V → ve (1.0)
c. D → un (1.0)
d. N → planeta (0.4)
e. N → telescopio (0.4)
f. P → con (1.0)
}
I = O

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	¿? [2,4]	[2,5]	[2,6]	¿? [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

$$[2,4] \rightarrow P(Regla\ 5) \times P(V) \times P(SN) = 0.7 \times 1.0 \times 0.4 = 0.28$$

Análisis Sintáctico - CKY

1. En la tabla CKY construida mediante el algoritmo CKY probabilístico para la secuencia “Ella ve un planeta con un telescopio”, falta por **calcular el valor de las celdas [2,4] y [2,7]**. ¿Qué **constituyentes** se han de colocar en ella y con qué valor de **probabilidad**?

T = {ella, ve, un, planeta, con, telescopio}
N = {O, SN, SV, D, N, P, Pron}
R = {
1. O → Pron SV (1.0)
2. SN → D N (1.0)
3. N → N SP (0.2)
4. SP → P SN (0.1)
5. SV→ V SN (0.7)
6. SV → SV SP (0.3)
a. Pron → ella (1.0)
b. V → ve (1.0)
c. D → un (1.0)
d. N → planeta (0.4)
e. N → telescopio (0.4)
f. P → con (1.0)
}
I = O

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	¿? [2,4]	[2,5]	[2,6]	¿? [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

$$[2,7] \rightarrow P(Regla\ 5) \times P(V) \times P(SN) = 0.7 \times 1.0 \times 0.0032 = 0.00224$$

Análisis Sintáctico - CKY

1. En la tabla CKY construida mediante el algoritmo CKY probabilístico para la secuencia “Ella ve un planeta con un telescopio”, falta por **calcular el valor de las celdas [2,4] y [2,7]**. ¿Qué **constituyentes** se han de colocar en ella y con qué valor de **probabilidad**?

T = {ella, ve, un, planeta, con, telescopio}
N = {O, SN, SV, D, N, P, Pron}
R = {
1. O → Pron SV (1.0)
2. SN → D N (1.0)
3. N → N SP (0.2)
4. SP → P SN (0.1)
5. SV→ V SN (0.7)
6. SV → SV SP (0.3)
a. Pron → ella (1.0)
b. V → ve (1.0)
c. D → un (1.0)
d. N → planeta (0.4)
e. N → telescopio (0.4)
f. P → con (1.0)
}
I = O

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	[2,4]	[2,5]	[2,6]	¿? [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

SV 0.28

$$[2,7] \rightarrow P(Regla\ 6) \times P(SV) \times P(SP) = 0.3 \times 0.28 \times 0.04 = 0.00336$$

Análisis Sintáctico - CKY

2. De acuerdo con el análisis obtenido arriba, ¿la oración “Ella ve un planeta con un telescopio” presenta ambigüedad sintáctica? Dibuja en forma de árbol sintáctico el que considere como preferido para esta oración de acuerdo con la tabla CKY.

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	SV 0.28 [2,4]	[2,5]	[2,6]	SV 0.00224 SV 0.00336 [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

- Hay ambigüedad sintáctica ya que existen dos posibles árboles sintácticos válidos tras aplicar el algoritmo CKY
- El de mayor probabilidad es el que corresponde a la agrupación:
  - O 0.00336

Análisis Sintáctico - CKY

2. De acuerdo con el análisis obtenido arriba, ¿la oración “Ella ve un planeta con un telescopio” presenta ambigüedad sintáctica? Dibuja en forma de árbol sintáctico el que considere como preferido para esta oración de acuerdo con la tabla CKY.

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	SV 0.28 [2,4]	[2,5]	[2,6]	SV 0.00224 SV 0.00336 [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

Análisis Sintáctico - CKY

2. De acuerdo con el análisis obtenido arriba, ¿la oración “Ella ve un planeta con un telescopio” presenta ambigüedad sintáctica? Dibuja en forma de árbol sintáctico el que considere como preferido para esta oración de acuerdo con la tabla CKY.

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	$\boxed{SV\ 0.28}$ [2,4]	[2,5]	[2,6]	SV 0.00224 $\boxed{SV\ 0.00336}$
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	$\boxed{\text{SP 0.04}}$ [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

Análisis Sintáctico - CKY

2. De acuerdo con el análisis obtenido arriba, ¿la oración “Ella ve un planeta con un telescopio” presenta ambigüedad sintáctica? Dibuja en forma de árbol sintáctico el que considere como preferido para esta oración de acuerdo con la tabla CKY.

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	SV 0.28 [2,4]	[2,5]	[2,6]	SV 0.00224 SV 0.00336 [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

Análisis Sintáctico - CKY

2. De acuerdo con el análisis obtenido arriba, ¿la oración “Ella ve un planeta con un telescopio” presenta ambigüedad sintáctica? Dibuja en forma de árbol sintáctico el que considere como preferido para esta oración de acuerdo con la tabla CKY.

Ella	ve	un	planeta	con	un	telescopio
Pron 1.0 [1,1]	[1,2]	[1,3]	[1,4]	[1,5]	[1,6]	O 0.00224 O 0.00336 [1,7]
	V 1.0 [2,2]	[2,3]	SV 0.28 [2,4]	[2,5]	[2,6]	SV 0.00224 SV 0.00336 [2,7]
		D 1.0 [3,3]	SN 0.4 [3,4]	[3,5]	[3,6]	SN 0.0032 [3,7]
			N 0.4 [4,4]	[4,5]	[4,6]	N 0.0032 [4,7]
				P 1.0 [5,5]	[5,6]	SP 0.04 [5,7]
					D 1.0 [6,6]	SN 0.4 [6,7]
						N 0.4 [7,7]

Semántica léxica - Lesk

Se necesita desambiguar el sentido de la palabra ‘banco’ en la siguiente frase:
“Trabajo la carpintería y la artesanía en el banco del taller.”

Aplica el algoritmo de Lesk original utilizando las siguientes definiciones de las palabras banco, carpintería, artesanía, taller, trabajar

Según el algoritmo ¿cuál es el sentido de dicha palabra en nuestra oración? Explica paso a paso la aplicación del algoritmo.
¿Es correcta la elección de significado del algoritmo?

banco1	Madero grueso escuadrado que se coloca horizontalmente sobre cuatro pies y sirve de mesa para labores de carpinteros y otros artesanos.
banco2	Conjunto de peces que van juntos en gran número.
carpintería	Taller o tienda donde trabaja el carpintero.
artesanía	Clase social constituida por los artesanos.
taller	Lugar en que se trabaja una obra de manos.
trabajar1	Dar o tomar algo por otra cosa que se considera del mismo o análogo valor.
tarbajar2	Ocuparse en cualquier actividad física o intelectual.

Semántica léxica - Lesk

- **Lesk Original**: Compara la firma de la palabra ambigua (signature), que se corresponde con su **definición en el diccionario**, con **las firmas de las palabras** de contexto (context), es decir, con las palabras vecinas a la palabra ambigua

1. Obtención de las firmas:

- Banco1: madero, horizontal, pie, mesa, labor, carpintero, artesano.
- Banco2: conjunto, pez, junto.
- Carpintería: taller, tienda, trabajar, carpintero.
- Artesanía: clase, social, artesano.
- Taller: lugar, trabajar, obra, mano.
- Trabajar1: dar, tomar, algo, cosa, mismo, valor.
- Trabajar2: ocupar, actividad, física, intelectual.

2. Matriz de coincidencias:

	banco1	banco2
Carpintería	1	0
Artesanía	1	0
Taller	0	0
Trabajar1	0	0
Trabajar2	0	0

Semántica léxica - Cálculo de similitud entre palabras

Semántica léxica - Cálculo de similitud entre palabras

1. Se utiliza la base de datos de relaciones léxicas WordNet para que un agente conversacional pueda interpretar diferentes palabras relacionadas con mobiliario. Al final del enunciado se muestran varias capturas de pantalla de los resultados que devuelve WordNet al buscar dos nombres de diferentes tipos de muebles: "table" (mesa), "chair" (silla). A partir de la información obtenida :

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

2. Calcule el valor de similitud basada en la longitud del camino entre table y chair. Explique el proceso y razone la respuesta usando el tesauro construido anteriormente.

Semántica léxica - Cálculo de similitud entre palabras

1. Se utiliza la base de datos de relaciones léxicas WordNet para que un agente conversacional pueda interpretar diferentes palabras relacionadas con mobiliario. Al final del enunciado se muestran varias capturas de pantalla de los resultados que devuelve WordNet al buscar dos nombres de diferentes tipos de muebles: "table" (mesa), "chair" (silla). A partir de la información obtenida :

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

2. Calcule el valor de similitud basada en la longitud del camino entre table y chair. Explique el proceso y razone la respuesta usando el tesauro construido anteriormente.

Semántica léxica - Cálculo de similitud entre palabras

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: table Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relation
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **table**, tabular array (a set of data arranged in rows and columns) *"se table 1"*
- S: (n) **table** (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs) *"it was a sturdy table"*
- S: (n) **table** (a piece of furniture with tableware for a meal laid out on it) *"I reserved a table at my favorite restaurant"*
- S: (n) mesa, **table** (flat tableland with steep edges) *"the tribe was relatively safe on the mesa but they had to descend into the valley for water"*
- S: (n) **table** (a company of people assembled at a table for a meal or game) *"he entertained the whole table with his witty remarks"*
- S: (n) board, **table** (food or meals in general) *"she sets a fine table"; "room and board"*

Word to search for: chair Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **chair** (a seat for one person, with a support for the back) *"he put his coat over the back of the chair and sat down"*
  - direct hyponym / full hyponym
  - part meronym
  - direct hypernym / inherited hypernym / sister term
    - S: (n) seat (furniture that is designed for sitting on) *"there were not enough seats for all the guests"*
      - direct hyponym / full hyponym
      - part meronym
      - member holonym
      - has instance
      - direct hypernym / inherited hypernym / sister term
        - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*
          - direct hyponym / full hyponym
          - domain term category
          - **direct hypernym** / inherited hypernym / sister term
            - S: (n) furnishing ((usually plural) the instrumentalities (furniture and appliances and other movable accessories including curtains and rugs) that make a home (or other area) livable)
          - derivationally related form
      - derivationally related form

Semántica léxica - Cálculo de similitud entre palabras

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

Noun

- S: (n) **table**, tabular array (a set of data arranged in rows and columns) *"see table 1"*
- S: (n) **table** (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs) *"it was a sturdy table"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *has instance*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*
      - *direct hyponym* / *full hyponym*
      - *domain term category*
      - *direct hypernym* / *inherited hypernym* / *sister term*
        - S: (n) furnishing ((usually plural) the instrumentalities (furniture and appliances and other movable accessories including curtains and rugs) that make a home (or other area) livable)
          - *direct hyponym* / *full hyponym*
          - *domain usage*
          - *direct hypernym* / *inherited hypernym* / *sister term*
            - S: (n) instrumentality, instrumentation (an artifact (or system of artifacts) that is instrumental in accomplishing some end)
              - *direct hyponym* / *full hyponym*
              - *direct hypernym* / *inherited hypernym* / *sister term*
                - S: (n) artifact, artefact (a man-made object taken as a whole)
                - *direct hyponym* / *full hyponym*
                - *direct hypernym* / *inherited hypernym* / *sister term*
                - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*

- S: (n) artifact, artefact (a man-made object taken as a whole)
  - *direct hyponym* / *full hyponym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
      - *direct hyponym* / *full hyponym*
      - *part meronym*
      - *direct hypernym* / *inherited hypernym* / *sister term*
        - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
          - *direct hyponym* / *full hyponym*
          - *direct hypernym* / *inherited hypernym* / *sister term*
            - S: (n) physical entity (an entity that has physical existence)
              - *direct hyponym* / *full hyponym*
              - *direct hypernym* / *inherited hypernym* / *sister term*
                - S: (n) **entity** (that

Semántica léxica - Cálculo de similitud entre palabras

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

Entity

Phisical Entity

Object

Unit

Artifact

S: (n) artifact, artefact (a man-made object taken as a whole)
- direct hyponym / full hyponym
- direct hypernym / inherited hypernym / sister term
  - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
    - direct hyponym / full hyponym
    - part meronym
    - direct hypernym / inherited hypernym / sister term
      - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
        - direct hyponym / full hyponym
        - direct hypernym / inherited hypernym / sister term
          - S: (n) physical entity (an entity that has physical existence)
            - direct hyponym / full hyponym
            - direct hypernym / inherited hypernym / sister term
              - S: (n) entity (that

Semántica léxica - Cálculo de similitud entre palabras

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

Entity

Phisical Entity

Object

Unit

Artifact

Instrumentality

Furnishing

Furniture

Table

Noun

- S: (n) **table**, tabular array (a set of data arranged in rows and columns) *"see table 1"*
- S: (n) **table** (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs) *"it was a sturdy table"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *has instance*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*
      - *direct hyponym* / *full hyponym*
      - *domain term category*
      - *direct hypernym* / *inherited hypernym* / *sister term*
        - S: (n) furnishing ((usually plural) the instrumentalities (furniture and appliances and other movable accessories including curtains and rugs) that make a home (or other area) livable)
          - *direct hyponym* / *full hyponym*
          - *domain usage*
          - *direct hypernym* / *inherited hypernym* / *sister term*
            - S: (n) instrumentality, instrumentation (an artifact (or system of artifacts) that is instrumental in accomplishing some end)
              - *direct hyponym* / *full hyponym*
              - *direct hypernym* / *inherited hypernym* / *sister term*
                - S: (n) artifact, artefact (a man-made object taken as a whole)
                - *direct hyponym* / *full hyponym*
                - *direct hypernym* / *inherited hypernym* / *sister term*
                - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*

Semántica léxica - Cálculo de similitud entre palabras

1. Construya un tesauro desde el nivel superior de la jerarquía hasta el nivel en el que aparezcan los dos nombres.

Word to search for: chair Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **chair** (a seat for one person, with a support for the back) *"he put his coat over the back of the chair and sat down"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) seat (furniture that is designed for sitting on) *"there were not enough seats for all the guests"*
      - *direct hyponym* / *full hyponym*
      - *part meronym*
      - *member holonym*
      - *has instance*
      - *direct hypernym* / *inherited hypernym* / *sister term*
        - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*
          - *direct hyponym* / *full hyponym*
          - *domain term category*
          - ***direct hypernym*** / *inherited hypernym* / *sister term*
            - S: (n) furnishing ((usually plural) the instrumentalities (furniture and appliances and other movable accessories including curtains and rugs) that make a home (or other area) livable)
          - *derivationally related form*
      - *derivationally related form*

Semántica léxica - Cálculo de similitud entre palabras

2. Calcule el valor de similitud basada en la longitud del camino entre table y chair. Explique el proceso y razone la respuesta usando el tesauro construido anteriormente.

Semántica léxica - Cálculo de similitud entre palabras

- **Longitud del camino entre dos sentidos**
  Se calcula sobre el gráfico del tesauro donde cada sentido es un nodo del mismo (*c1, c2*)
  De manera que *pathlen(c1,c2)* = #aristas(shortest_path(c1,c2)) +1

- Así, la medida de **similitud basada en la longitud del camino** queda:

$$sim_{path(c_1,c_2)} = \frac{1}{pathlen(c_1, c_2)}$$

$$wordsim(w_1, w_2) = \max_{\substack{c_1 \in senses(w_1) \\ c_2 \in senses(w_2)}} sim(c_1, c_2)$$

Semántica léxica - Cálculo de similitud entre palabras

2. Calcule el valor de similitud basada en la longitud del camino entre table y chair. Explique el proceso y razone la respuesta usando el tesauro construido anteriormente.

pathlen(chair, table) = 3 +1 = 4

sim (chair, table) = 1/ pathlen(chair, table) = 1/4 = 0.25

Modelos de representación vectorial

Dadas las tres siguientes oraciones:

- O1: *El apartamento que alquilé es el lugar más acogedor que he tenido.*
- O2*: Compré bananas porque no quedaba más fruta en la feria.*
- O3: *La leyenda de Prometeo y el robo del fuego es mi mito favorito.*

Sobre ellas, se ha realizado un proceso de tokenización, eliminación stop-words y lematización.
El resultado es:

- ON1: apartamento, alquilar, lugar, acogedor
- ON2: comprar, banana, quedar, fruta, feria
- ON3: leyenda, prometeo, robo, fuego, mito, favorito

Tras ello, se construyen diferentes versiones de Bag of Words para ON1, ON2 y ON3.
Se aporta, como parte del enunciado, los vectores BoW resultantes de aplicar el criterio binario.
Se pide:

1. Construye los vectores para los métodos TF, WTF y TF-IDF, según las definiciones de éstos dadas en los materiales de la asignatura. Explica el procedimiento que has seguido. Para los cálculos del TF-IFD usa el logaritmo en base 10.

Matriz TF = cuenta el número de veces que aprecen los terminos en la oración
Matriz WTF = normaliza los valores de las relaciones respecto a su número de palabras
Matriz TF-IDF = Utiliza información relacionada a los otros textos o documentos

2. Considerando que dos oraciones son más similares entre sí cuanto menor es su distancia euclídea, indica que par de oraciones son más similares entre sí. Haz el cálculo para los casos TF y WTF. Explica el procedimiento que has seguido.

3. Explica qué significa el valor obtenido para ‘alquilar’ en el criterio TF-IDF.

Modelos de representación vectorial

	ON1	ON2	ON3
acogedor	1	0	0
alquilar	1	0	0
apartamento	1	0	0
banana	0	1	0
comprar	0	1	0
favorito	0	0	1
feria	0	1	0
fruta	0	1	0
fuego	0	0	1
leyenda	0	0	1
lugar	1	0	0
mito	0	0	1
prometeo	0	0	1
quedar	0	1	0
robo	0	0	1

BoW binario

Modelos de representación vectorial

	ON1	ON2	ON3
acogedor	1	0	0
alquilar	1	0	0
apartamento	1	0	0
banana	0	1	0
comprar	0	1	0
favorito	0	0	1
feria	0	1	0
fruta	0	1	0
fuego	0	0	1
leyenda	0	0	1
lugar	1	0	0
mito	0	0	1
prometeo	0	0	1
quedar	0	1	0
robo	0	0	1

Matriz TF

Modelos de representación vectorial

	ON1	ON2	ON3
acogedor	0.25	0.0	0.000
alquilar	0.25	0.0	0.000
apartamento	0.25	0.0	0.000
banana	0.00	0.2	0.000
comprar	0.00	0.2	0.000
favorito	0.00	0.0	0.167
feria	0.00	0.2	0.000
fruta	0.00	0.2	0.000
fuego	0.00	0.0	0.167
leyenda	0.00	0.0	0.167
lugar	0.25	0.0	0.000
mito	0.00	0.0	0.167
prometeo	0.00	0.0	0.167
quedar	0.00	0.2	0.000
robo	0.00	0.0	0.167

Matriz WTF

Ejemplos:

- acogedor -> 1/4 = 0.25
- banana -> 1/5 = 0.2
- favorito -> 1/6 = 0.167

Modelos de representación vectorial

	ON1	ON2	ON3
acogedor	0.477	0.000	0.000
alquilar	0.477	0.000	0.000
apartamento	0.477	0.000	0.000
banana	0.000	0.477	0.000
comprar	0.000	0.477	0.000
favorito	0.000	0.000	0.477
feria	0.000	0.477	0.000
fruta	0.000	0.477	0.000
fuego	0.000	0.000	0.477
leyenda	0.000	0.000	0.477
lugar	0.477	0.000	0.000
mito	0.000	0.000	0.477
prometeo	0.000	0.000	0.477
quedar	0.000	0.477	0.000
robo	0.000	0.000	0.477

Matriz TF-IDF

Modelos de representación vectorial

$$\text{Distancia Euclídea} = \sqrt{\sum_{i=1}^{N}(X_{i1} - X_{i2})^2}$$

Distancia euclidea (TF) entre oraciones 1 y 2: 3.0
Distancia euclidea (TF) entre oraciones 1 y 3: 3.1623
Distancia euclidea (TF) entre oraciones 2 y 2: 3.3166

Distancia euclidea (WTF) entre oraciones 1 y 2: 0.6708
Distancia euclidea (WTF) entre oraciones 1 y 3: 0.6455
Distancia euclidea (WTF) entre oraciones 2 y 3: 0.6056

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 6 - Modelado del Lenguaje Neuronal

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

En el día de hoy

¿Cómo podemos procesar un texto de gran tamaño con una red neuronal?

Encuesta previa

- ¿Qué es relevante en una red recurrente?
- ¿Qué diferencia una RNN de una LSTM?
- ¿Cuáles son los tipos de arquitecturas de RNN que hay?
- ¿Qué es un embedding contextual?

En el día de hoy

- embeddings, RNN, LSTM (recordatorio)

- Transformers
  - Atención
  - Posicionamiento

- Modelos preentrenados
  - BERT
  - BETO
  - HuggingFace

Embeddings (recordatorio)

- Word2Vec

- Vectores que representan palabras, y guardan relación con su significado

- Generados con entrenamiento no supervisado

RNN y LSTM (recordatorio)

ELMO (recordatorio)

- Utiliza redes **LSTM bidireccionales** para extraer el embedding
  - El embedding hace mejor uso del contexto

- Una palabra no tiene representación única
  - Un embedding diferente **para una misma palabra** en oraciones diferentes

Transformers

Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). **Attention is all you need**. *Advances in neural information processing systems*, *30*.

Attention is all you need

A Vaswani, N Shazeer, N Parmar... - Advances in neural ..., 2017 - proceedings.neurips.cc

... to attend to **all** positions in the decoder up to and including that position. **We need** to prevent ... **We** implement this inside of scaled dot-product **attention** by masking out (setting to $-\infty$) ...

Save Cite Cited by 123911 Related articles All 87 versions

Dato a 09/06/2024

Transformers

Transformers. Dos conceptos clave

- Atención
- Embeddings posicionales

Atención

Conceptos

- **Query**: representa el elemento de interés a analizar (e.j., “it”)
- **Key**: Sirve para ver cómo de relevantes son el resto de elementos de la secuencia de entrada con respecto al element de interés (e.j., “animal”, “street”…). El *“score” de atención* se obtiene en función de query-key
- **Values**: Representa, como “key”, los elementos de la secuencia, que se ponderarán en función del *score de atención*

https://jalammar.github.io/illustrated-transformer/

Atención

y_3
5. Output Vector
4. Weight and Sum value vectors
3. Softmax $\alpha_{i,j}$
2. Key/Query Comparisons
1. Generate key, query value vectors
x_1
x_2
x_3
k
q
v

Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Atención: 1. Generar vectores q, k, v

```
Weights for key:
 tensor([[0., 0., 1.],
         [1., 1., 0.],
         [0., 1., 0.],
         [1., 1., 0.]])
Weights for query:
 tensor([[1., 0., 1.],
         [1., 0., 0.],
         [0., 0., 1.],
         [0., 1., 1.]])
Weights for value:
 tensor([[0., 2., 0.],
         [0., 3., 0.],
         [1., 0., 3.],
         [1., 1., 0.]])
```

Self-attention

Matriz de pesos Q, K, V

- Inicialmente, valores aleatorios
- En el entrenamiento, se ajustarán sus valores

Ejemplo de vector de entrada a la capa de Atención

https://colab.research.google.com/drive/1rPk3ohrmVclqhH7uQ7qys4oznDdAhpzF#scrollTo=jUTNr15JBkSG

Atención: 1. Generar vectores q, k, v

```
Weights for key:
 tensor([[0., 0., 1.],
         [1., 1., 0.],
         [0., 1., 0.],
         [1., 1., 0.]])
Weights for query:
 tensor([[1., 0., 1.],
         [1., 0., 0.],
         [0., 0., 1.],
         [0., 1., 1.]])
Weights for value:
 tensor([[0., 2., 0.],
         [0., 3., 0.],
         [1., 0., 3.],
         [1., 1., 0.]])
```

Self-attention

Obtenemos la “key” multiplicando los vectores de entrada por la matriz de pesos K

Keys -> x * W_key

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} = \boxed{\begin{pmatrix} 0 & 1 & 1 \\ 4 & 4 & 0 \\ 2 & 3 & 1 \end{pmatrix}}$$

Atención: 1. Generar vectores q, k, v

```
Weights for key:
 tensor([[0., 0., 1.],
         [1., 1., 0.],
         [0., 1., 0.],
         [1., 1., 0.]])
Weights for query:
 tensor([[1., 0., 1.],
         [1., 0., 0.],
         [0., 0., 1.],
         [0., 1., 1.]])
Weights for value:
 tensor([[0., 2., 0.],
         [0., 3., 0.],
         [1., 0., 3.],
         [1., 1., 0.]])
```

Self-attention

Mismo proceso para obtener la salida de "value" -> x * W_value

Atención: 1. Generar vectores q, k, v

```
Weights for key:
 tensor([[0., 0., 1.],
         [1., 1., 0.],
         [0., 1., 0.],
         [1., 1., 0.]])
Weights for query:
 tensor([[1., 0., 1.],
         [1., 0., 0.],
         [0., 0., 1.],
         [0., 1., 1.]])
Weights for value:
 tensor([[0., 2., 0.],
         [0., 3., 0.],
         [1., 0., 3.],
         [1., 1., 0.]])
```

Mismo proceso para obtener la salida de "query" -> x * W_query

Atención: 2. Comparar key/queries

$$QK^T$$

	q1•k1	q1•k2	q1•k3
	q2•k1	q2•k2	q2•k3
N	q3•k1	q3•k2	q3•k3

$$\begin{pmatrix} 1 & 0 & 2 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = 2$$

Atención: 3. Capa softmax

$$softmax(QK^T)$$

Softmax function

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax

$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector

Nota: Los valores están redondeados a 1 decimal por simplicidad -> [0.06, 0.47, 0.47] => [0, 0.5, 0.5]

Atención: 4. Multiplicar por el vector values

Atención: 5. Sumar las salidas y obtener output

Atención: Mismo proceso para output 2

Atención: Mismo proceso para output 3

Bloque de Transformer

Multihead attention

Embeddings posicionales

- Se combinan los embeddings sobre las palabras de entrada con un embedding adicional que tiene en cuenta su posición en la secuencia

Embeddings posicionales

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

- **k**: posición del token en la secuencia
- **d**: dimension de los embeddings (a definir por el usuario)
- **n**: valor escalar (a definer por el usuario); vale 10k en la arquitectura original de Transformers

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Embeddings posicionales

- https://mfaizan.github.io/2023/04/02/sines.html
- https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
- https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

BERT

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Modelo de Lenguaje basado en **Transformers**

- Pre-entrenado con BooksCorpus (800M words) y Wikipedia en inglés (2,500M words)
- Genera embeddings contextuales
- Entrenable para multiples tareas

BERT. Preentrenamiento

- Pre-entrenado con BooksCorpus (800M words) y Wikipedia en inglés (2,500M words)
  - Masked LM (MLM)
  - Next Sentence Prediction (NSP)

BERT demo

- Ejemplo extraído de https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

https://colab.research.google.com/drive/1ZQvuAVwA3IjybezQOXnrXMGAnMyZRuPU

BETO: Spanish BERT

- BETO is a BERT model trained on a big Spanish corpus.

Canete, J., Chaperon, G., Fuentes, R., Ho, J. H., Kang, H., & Pérez, J. (2020). **Spanish pre-trained bert model and evaluation data**. *Pml4dc at iclr*, *2020*, 1-10.

BETO: Spanish BERT

- *Spanish Wikis*: Wich include Wikipedia, Wikinews, Wikiquotes and more.
- *ParaCrawl*: Spanish portion of ParaCrawl (http://opus.nlpl.eu/ParaCrawl.php)
- *EUBookshop*: Spanish portion of EUBookshop (http://opus.nlpl.eu/EUbookshop.php)
- *MultiUN*: Spanish portion of MultiUN (http://opus.nlpl.eu/MultiUN.php)
- *OpenSubtitles*: Spanish portion of OpenSubtitles2018 (http://opus.nlpl.eu/OpenSubtitles-v2018.php)
- *DGC*: Spanish portion of DGT (http://opus.nlpl.eu/DGT.php)
- *DOGC*: Spanish portion of DOGC (http://opus.nlpl.eu/DOGC.php)
- *ECB*: Spanish portion of ECB (http://opus.nlpl.eu/ECB.php)
- *EMEA*: Spanish portion of EMEA (http://opus.nlpl.eu/EMEA.php)
- *Europarl*: Spanish portion of Europarl (http://opus.nlpl.eu/Europarl.php)
- *GlobalVoices*: Spanish portion of GlobalVoices (http://opus.nlpl.eu/GlobalVoices.php)
- *JRC*: Spanish portion of JRC (http://opus.nlpl.eu/JRC-Acquis.php)
- *News-Commentary11*: Spanish portion of NCv11 (http://opus.nlpl.eu/News-Commentary-v11.php)
- *TED*: Spanish portion of TED (http://opus.nlpl.eu/TED2013.php)
- *UN*: Spanish portion of UN (http://opus.nlpl.eu/UN.php)

Más modelos en HuggingFace

https://huggingface.co/models

Dato a 09/06/2024

HuggingFace

Librería Transformers

https://huggingface.co/docs/transformers/index

Algunos notebooks útiles

- **HuggingFace Pipeline**
- **HuggingFace FineTuning**

En resumen

- Los Transformers evolucionan sobre LSTM y RNN con:
  - Atención
  - Encoding posicional
- Diferentes configuraciones de Transformers dan lugar a modelos de lenguaje neuronales **preentrenados**.
- **HuggingFace** es una comunidad que contiene múltiples modelos.

En la próxima semana

¿Cómo utilizar grandes modelos de lenguaje para pre-entrenamiento y fine-tuning?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 9 – Aplicaciones de NLP

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

En el día de hoy…

- Recordatorio de tecnologías clave

- **Generación de texto**
- **Autocompletado de texto**
- **Generación automática de resúmenes**
- Traducción automática
- Question Answering
- Análisis de sentimientos
- Otras

Tecnologías clave: Word embeddings

- Vectores que representan palabras, y guardan relación con su significado
- Generados con entrenamiento no supervisado

Redes recurrentes

Fang, Hao & Jiang, Man & Lu, Jingyan & Qin, Gong & Tao, & Tang, Peifu & Yunjun, Yan & Yun, Luo & Liu, Jingtian. (2020). A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages. Remote Sensing. 12. 256. 10.3390/rs12020256.

Tecnologías clave: Encoder-Decoder (seq2seq)

- Vector encoder (o “contexto”)

Transformers

- Atención
- Embeddings posicionales

https://kikaben.com/transformers-encoder-decoder/

Transformers: Causal vs No Causal

$$SelfAttention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Self-attention causal

	q1•k1	−∞	−∞	−∞	−∞
	q2•k1	q2•k2	−∞	−∞	−∞
N	q3•k1	q3•k2	q3•k3	−∞	−∞
	q4•k1	q4•k2	q4•k3	q4•k4	−∞
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

Self-attention bidireccional

	q1•k1	q1•k2	q1•k3	q1•k4	q1•k5
	q2•k1	q2•k2	q2•k3	q2•k4	q2•k5
N	q3•k1	q3•k2	q3•k3	q3•k4	q3•k5
	q4•k1	q4•k2	q4•k3	q4•k4	q4•k5
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

RNN: Causal vs No Causal

RNN Causal: Unidireccional

RNN No Causal: Bidireccional

Tecnologías clave: Distintos tipos de arquitecturas

Generación te textos

- **Autosupervisados**
- **Teacher Forcing**

Generación de textos

- **Sampling** para elegir el siguiente token en base a la distribución de probabilidad predicha

Generación de textos

$$
\begin{aligned}
& \mathrm{i} \leftarrow 1 \\
& w_i \sim \mathrm{p}(w) \\
& \textbf{while } w_i \text{ != EOS} \\
& \quad \mathrm{i} \leftarrow \mathrm{i} + 1 \\
& \quad w_i \sim \mathrm{p}(w_i \mid w_{<i})
\end{aligned}
$$

- Con ese **muestreo**, se van generando tokens nuevos, eligiendo el que se obtenga del muestreo/sampling sobre la distribución predicha
- En cada iteración, la distribución de probabilidad estará **condicionada** a la información generada hasta ese momento.
- Se **termina** de generar tokens cuando se obtenga un token tipo EOS

Generación de textos: otros muestreos

- ***Top-k sampling***: En vez de usar toda la distribución de probabilidad del vocabulario, se eligen primero los **k tokens con mayor probabilidad**, y se hace el muestreo sobre ellos.
  - 1) Se elige un valor de k
  - 2) Se obtiene la probabilidad del vocabulario y se seleccionan los tokens con mayor probabilidad
  - 3) Se normalizan de nuevo las probabilidades sobre ese subconjunto
  - 4) Se hace el muestreo para elegir el siguiente token

- ***Top-p sampling:*** En lugar de seleccionar los k tokens más probables, se seleccionan los tokens que pertenecen **top del porcentaje p** de mayor probabilidad. Así, corresponden al menor número de tokens que cumplen esta condición.

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p.$$

Generación de textos: otros muestreos

- ***Temperature sampling***: En vez de seleccionar subconjuntos de tokens, se reajusta la distribución de probabilidad, para alterar el muestreo. Se dividen los valores de salida (antes de aplicar la función softmax) por un parámetro τ.

$$\mathbf{y} = \mathrm{softmax}(u) \quad \Rightarrow \quad \mathbf{y} = \mathrm{softmax}(u/\tau)$$

- Si $\tau = 1$, no hay cambios.
- Cuando más cercano sea a 0, más se subirá la probabilidad de la palabra con mayor valor (que será más cercana a 1). De esta manera, habrá **menos variabilidad** en la secuencia generada.
- Cuando mayor sea el valor (por encima de 1) más se igualarán las probabilidades de la distribución, y habrá **más variabilidad** en la secuencia generada

Referencia bibliográfica clave

- **Speech and Language Processing (3rd ed. draft)**

  Dan Jurafsky and James H. Martin

- https://web.stanford.edu/~jurafsky/slp3/

Aplicaciones: Autocompletado de texto

```
Las cosas podían haber sucedido...

... de cualquier otra manera y, sin embargo, sucedieron así.

... de la forma más maravillosa.
... aquel martes, aunque tuvimos que esperar dos años para que sucedieran.
... tal y como dijiste.
```

- ¿Qué tecnologías pueden realizar esta tarea?

Aplicaciones: Autocompletado de textos

- **Objetivo**: Dada una secuencia de texto de entrada, generar cómo podría continuar dicha secuencia

Seq2Seq

Aplicaciones: Generación de resúmenes

- **Objetivo**: generar un texto más corto que el original y que condense el significado principal de éste
- **Datos de entrenamiento**: pares de textos originales – textos resumidos
- **Procesado**: Se combinan ambas secuencias, delimitándolas por un token fijo (e.j., δ)
- **Entrenamiento**: se tratarán de predecir ambas secuencias como una única
- **Inferencia**: Se introduce el texto a resumir, el sistema le concatena el token δ, y se busca predecir el resto de tokens, que corresponderían al resumen

Secuencia original: $(x_1, \dots, x_m)$

Resumen: $(y_1, \dots, y_n)$

Secuencia de entrada: $(x_1, \dots, x_m, \delta, y_1, \dots, y_n)$

Aplicaciones: Generación de resúmenes

Secuencia original: $(x_1, \dots, x_m)$

Resumen: $(y_1, \dots, y_n)$

Secuencia de entrada: $(x_1, \dots, x_m, \delta, y_1, \dots, y_n)$

Recursos

13 - aplicacionesNLP.ipynb

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Alberto Barbado González

Tema 9 – Aplicaciones de NLP

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

En el día de hoy…

- Recordatorio de tecnologías clave

- Generación de texto
- Autocompletado de texto
- Generación automática de resúmenes
- Traducción automática
- Question Answering
- Análisis de sentimientos
- Otras

Tecnologías clave: Word embeddings

- Vectores que representan palabras, y guardan relación con su significado
- Generados con entrenamiento no supervisado

Redes recurrentes

Fang, Hao & Jiang, Man & Lu, Jingyan & Qin, Gong & Tao, & Tang, Peifu & Yunjun, Yan & Yun, Luo & Liu, Jingtian. (2020). A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages. Remote Sensing. 12. 256. 10.3390/rs12020256.

Tecnologías clave: Encoder-Decoder (seq2seq)

- Vector encoder (o “contexto”)

Transformers

- Atención
- Embeddings posicionales

https://kikaben.com/transformers-encoder-decoder/

Transformers: Causal vs No Causal

$$SelfAttention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Self-attention causal

	q1•k1	−∞	−∞	−∞	−∞
	q2•k1	q2•k2	−∞	−∞	−∞
N	q3•k1	q3•k2	q3•k3	−∞	−∞
	q4•k1	q4•k2	q4•k3	q4•k4	−∞
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

Self-attention bidireccional

	q1•k1	q1•k2	q1•k3	q1•k4	q1•k5
	q2•k1	q2•k2	q2•k3	q2•k4	q2•k5
N	q3•k1	q3•k2	q3•k3	q3•k4	q3•k5
	q4•k1	q4•k2	q4•k3	q4•k4	q4•k5
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

RNN: Causal vs No Causal

RNN Causal: Unidireccional

RNN No Causal: Bidireccional

Tecnologías clave: Distintos tipos de arquitecturas

Generación te textos

- **Autosupervisados**
- **Teacher Forcing**

Generación de textos

- **Sampling** para elegir el siguiente token en base a la distribución de probabilidad predicha

Generación de textos

$$
\begin{aligned}
& \mathrm{i} \leftarrow 1 \\
& w_i \sim \mathrm{p}(w) \\
& \textbf{while } w_i \text{ != EOS} \\
& \quad \mathrm{i} \leftarrow \mathrm{i} + 1 \\
& \quad w_i \sim \mathrm{p}(w_i \mid w_{<i})
\end{aligned}
$$

- Con ese **muestreo**, se van generando tokens nuevos, eligiendo el que se obtenga del muestreo/sampling sobre la distribución predicha
- En cada iteración, la distribución de probabilidad estará **condicionada** a la información generada hasta ese momento.
- Se **termina** de generar tokens cuando se obtenga un token tipo EOS

Generación de textos: otros muestreos

- ***Top-k sampling***: En vez de usar toda la distribución de probabilidad del vocabulario, se eligen primero los **k tokens con mayor probabilidad**, y se hace el muestreo sobre ellos.
  - 1) Se elige un valor de k
  - 2) Se obtiene la probabilidad del vocabulario y se seleccionan los tokens con mayor probabilidad
  - 3) Se normalizan de nuevo las probabilidades sobre ese subconjunto
  - 4) Se hace el muestreo para elegir el siguiente token

- ***Top-p sampling:*** En lugar de seleccionar los k tokens más probables, se seleccionan los tokens que pertenecen **top del porcentaje p** de mayor probabilidad. Así, corresponden al menor número de tokens que cumplen esta condición.

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p.$$

Generación de textos: otros muestreos

- ***Temperature sampling***: En vez de seleccionar subconjuntos de tokens, se reajusta la distribución de probabilidad, para alterar el muestreo. Se dividen los valores de salida (antes de aplicar la función softmax) por un parámetro τ.

$$\mathbf{y} = \mathrm{softmax}(u) \quad \Rightarrow \quad \mathbf{y} = \mathrm{softmax}(u/\tau)$$

- Si $\tau = 1$, no hay cambios.
- Cuando más cercano sea a 0, más se subirá la probabilidad de la palabra con mayor valor (que será más cercana a 1). De esta manera, habrá **menos variabilidad** en la secuencia generada.
- Cuando mayor sea el valor (por encima de 1) más se igualarán las probabilidades de la distribución, y habrá **más variabilidad** en la secuencia generada

Referencia bibliográfica clave

- **Speech and Language Processing (3rd ed. draft)**

  Dan Jurafsky and James H. Martin

- https://web.stanford.edu/~jurafsky/slp3/

Aplicaciones: Autocompletado de texto

```
Las cosas podían haber sucedido...

... de cualquier otra manera y, sin embargo, sucedieron así.

... de la forma más maravillosa.
... aquel martes, aunque tuvimos que esperar dos años para que sucedieran.
... tal y como dijiste.
```

- ¿Qué tecnologías pueden realizar esta tarea?

Aplicaciones: Autocompletado de textos

- **Objetivo**: Dada una secuencia de texto de entrada, generar cómo podría continuar dicha secuencia

Seq2Seq

Aplicaciones: Generación de resúmenes

- **Objetivo**: generar un texto más corto que el original y que condense el significado principal de éste
- **Datos de entrenamiento**: pares de textos originales – textos resumidos
- **Procesado**: Se combinan ambas secuencias, delimitándolas por un token fijo (e.j., δ)
- **Entrenamiento**: se tratarán de predecir ambas secuencias como una única
- **Inferencia**: Se introduce el texto a resumir, el sistema le concatena el token δ, y se busca predecir el resto de tokens, que corresponderían al resumen

Secuencia original: $(x_1, \dots, x_m)$

Resumen: $(y_1, \dots, y_n)$

Secuencia de entrada: $(x_1, \dots, x_m, \delta, y_1, \dots, y_n)$

Aplicaciones: Generación de resúmenes

Secuencia original: $(x_1, \dots, x_m)$

Resumen: $(y_1, \dots, y_n)$

Secuencia de entrada: $(x_1, \dots, x_m, \delta, y_1, \dots, y_n)$

Recursos

13 - aplicacionesNLP.ipynb

Aplicaciones: traducción automática

- Objetivo: Dado un texto en un idioma, obtener un texto en otro idioma con el mismo significado

Aplicaciones: traducción automática

- Evaluación del modelo: fidelidad
  - **Word error rate (WER):** Dividir el número de errores en las palabras de la frase traducida con respecto al número de palabras totales
    - S: Número de sustituciones
    - D: Número de eliminaciones
    - I: Número de inserciones
    - C: Número de palabras correctas
    - N = S + D + C -> total de palabras en la referencia

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

- Referencia: “amidst the emerald meadow, butterflies whispered”
- Predicción: “amidst the emerald shadow, butterflies danced”

C = 4
D = 0
I = 0
S = 2

$$WER = \frac{2 + 0 + 0}{2 + 0 + 0 + 4} = 0.33$$

Aplicaciones: traducción automática

- Evaluación del modelo: fidelidad
  - ***character F-score*** (chrF), que compara el solape en n-gramas que hay entre la frase traducida con el modelo de MT y la traducción original.
    - Referencia: “amidst the emerald meadow, butterflies whispered”
    - Predicción: “amidst the emerald shadow, butterflies danced”

$$\text{CER} = \frac{\text{Substitutions} + \text{Deletions} + \text{Insertions}}{\#\text{ of Characters in Reference}}$$

Aplicaciones: traducción automática

- Evaluación del modelo: fidelidad
  - **BLEU (*B*i*L*ingual *E*valuation *U*nderstudy):** Cuantifica la similitud entre el texto candidato y el texto de referencia como una puntuación entre 0 y 1. 0 significa que no hay superposición con el texto real y 1 significa que hay una superposición perfecta.

$$\begin{aligned}\text{BLEU} &= \text{Brevity Penalty} \times \text{n-gram Overlap} \\ &= \min\left(1, \exp\left(1 - \frac{\text{reference length}}{\text{output length}}\right)\right) \times \prod_{i=1}^{4} \text{i-gram Precision}^{\frac{1}{4}}\end{aligned}$$

  - **BERTScore,** con la que se obtienen los *embeddings* de las palabras de la traducción de referencia y las de la traducción obtenida, y se comparan ambos vectores.

Aplicaciones: traducción automática

Referencias

https://docs.kolena.com/metrics/wer-cer-mer/

https://docs.kolena.com/metrics/bleu/

https://docs.kolena.com/metrics/bertscore/

https://docs.kolena.com/metrics/meteor/

https://docs.kolena.com/metrics/rouge-n/

Aplicaciones: Question Answering

- Objetivo: responder a preguntas que plantea el usuario

  (asumimos que nos limitamos a preguntas sobre un texto o corpus)

  - Tipos de preguntas
    - Respuesta Si/No:
      - *¿La noche de San Juan va del 23 al 24 de junio?*
    - Respuesta cerrada
      - *¿Cuándo es la noche de San Juan?*
    - Respuesta abierta
      - ¿Cómo se celebra la noche de San Juan?

  - Tipos de estrategias
    - Information Retrieval
    - Information Extraction / Response Generation

Aplicaciones: Question Answering

- Information Retrieval

- Se devuelve el documento encontrado, tal y como se encuentra en el repositorio

Aplicaciones: Question Answering

- Information Retrieval

- Un aspecto relevante es cómo se realiza la búsqueda
  - ¿Qué técnicas podemos usar para realizar la búsqueda?

e.j. ->

$$\text{score}(q,d) = \sum_{t \in q} \frac{\text{tf-idf}(t,d)}{|d|}$$

Aplicaciones: Question Answering

- Information Extraction / Response Generation

Aplicaciones: Question Answering

- Retreival Augmented Generation (RAG)

https://ai-for-production.blog/2024/01/24/understanding-rag-and-vector-db/

Aplicaciones: Question Answering

- Information Extraction / Response Generation

Evaluación => $$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

Aplicaciones: Análisis de sentimientos

- Objetivo: etiquetar un texto de acuerdo a la emoción que subyace de él
  - Tipos de análisis:
    - De polaridad
    - De emociones básicas

Aplicaciones: Análisis de sentimientos

- Análisis de polaridad
- Análisis de emociones

Aplicaciones: Análisis de sentimientos

https://www.mdpi.com/2079-9292/9/3/483

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Presentación de la asignatura

LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

Agenda

- Presentaciones
- Objetivos de la asignatura
- Contenidos y estructura
- Actividades puntuables
- Evaluación

Presentaciones

Universidad Internacional de La Rioja

Acerca de…

Luis de la Fuente Valentín

- Doctorado en Ingeniería Telemática (es decir, perfil técnico)
- Perfil de investigación: Learning Analytics, Analítica de Datos, Procesado de Lenguaje Natural

Acerca de…

Otros profesores en la asignatura

- Alberto Barbado
- Yerai Mezquita Martín
- Javier Hernández Afonso

Acerca de…

Vosotros

- ¿Desde qué ciudad te conectas?
- ¿Qué esperas de esta asignatura?
- Una palabra que relaciones con PLN

Objetivos de la asignatura

Objetivos de aprendizaje

- **Conceptos Básicos**
  - Conocer el origen y la evolución del PLN.
  - Preprocesar un texto para su tratamiento con PLN
  - Reconocer los principales recursos para el uso de PLN.
- **PLN basado en ligüística**
  - Entender la relación entre lingüística y PLN.
  - Realizar análisis morfológico / sintáctico / semántico
- **PLN estadístico y neuronal**
  - Construir un predictor probabilístico de texto.
  - Representar palabras a través de su forma vectorial.
  - Utilizar redes neuronales capaces de procesar secuencias.
  - Utilizar redes neuronales tipo transformers.
  - Reconocer los conceptos de preentrenamiento y fine-tuning
- **Aplicaciones**
  - Reconocer las principales aplicaciones del PLN.
  - Reconocer las técnicas que se utilizan en las principales aplicaciones del PLN.

Contenidos y estructura

Sobre la asignatura: contenido

- Bloque I
  - T1: Introducción
  - T2: El Texto como Dato
- Bloque II
  - T3: Análisis Morfosintáctico
  - T4: Análisis Sintáctico
  - T5: Análisis Semántico
  - T6: Semántica Léxica
- Bloque III
  - T7: Modelado del Lenguaje
  - T8: Modelado de Lenguaje Neuronal
- Bloque IV
  - T9: Aplicaciones del PLN
  - T10: Agentes conversacionales

Sobre la asignatura: planificación

Actividades puntuables

Actividades

- Proyecto transversal: Caracterización de textos
- Laboratorio: Word Embedings y Transformers para clasificación de texto
- Actividad grupal: Aplicaciones prácticas de PLN en investigación y en el mercado final

Test de autoevaluación

- Hay un test en cada tema

- El objetivo es que sirvan de repaso, pero también de incentivo para el estudio

Actividades no puntuables

Foro de consultas

- NO puntúa
- Se agradece la participación
- Tiempo de respuesta aproximado: 24 horas

Asistencia a clases

- NO puntúa
- Posibilidad de asistencia en directo y visualización en diferido
- Preguntas a través del chat
- Las transparencias de las sesiones se colocarán en la sección Documentación

Evaluación

Sobre la asignatura: evaluación

- Evaluación continua (40%)
  - Actividades
    - Trabajo Individual (5)
    - Laboratorio (5)
    - Actividad grupal (4)

    Las actividades no aprobadas, no suman

    - Test de autoevaluación (0.1 x 10)

- Examen (60%)

  Es obligatorio aprobar el examen

Sobre la asignatura: evaluación

```
if (notaExamen >= 5) {

    notaContinua = min(10, notaAcumuladaEnContinua);
    notaFinal = (notaContinua * 0,4) + (notaExamen * 0,6);

} else {

    notaFinal = notaExamen * 0,6;

}
```

unir

LA UNIVERSIDAD
EN INTERNET

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Bloque I - Tema 1 - Introducción

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Problema del día

- ¿Qué características concretas debe tener una IA para considerarse Procesamiento de Lenguaje Natural?

Encuesta previa

- Nombra una **herramienta** que conozcas que usa PLN

- Nombra una **tarea** que conozcas que se realiza con PLN

En el día de hoy

- Definición de Procesamiento de Lenguaje Natural
- Aplicaciones de PLN
- Lingüística y PLN
- Historia del Procesamiento de Lenguaje Natural

Definición de Procesamiento de Lenguaje Natural

Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do. (IBM).

El procesamiento de lenguaje natural es un campo de las ciencias de la computación, de la inteligencia artificial y de la lingüística que estudia las interacciones entre las computadoras y el lenguaje humano. (Wikipedia)

Campo de la Inteligencia Artificial que tiene como objetivo principal hacer que las máquinas sean capaces de realizar tareas que involucren el lenguaje humano (Apuntes de la asignatura)

Otros Nombres

- Procesamiento de Lenguaje Natural (PLN)
- Natural Language Processing (NLP)
- Procesamiento del Lenguaje y el Habla
- Tecnología del Lenguaje
- Lingüística computacional
- Reconocimiento y Síntesis del Habla

Aplicaciones de PLN

- Information Retrieval
- Question Answering
- Resumen Automático
- Traducción Automática
- Generación de Textos
- Language Modeling
- Named Entity Recognition
- Análisis Sintáctico
- Análisis Morfosintáctico (POS Tag)
- Word Sense Disambiguation
- Speech Recognition
- Speech Generation
- Sistemas de Diálogo

Aplicaciones de PLN

- Information Retrieval
- **Question Answering**
- Resumen Automático
- **Traducción Automática**
- Generación de Textos
- Language Modeling
- Named Entity Recognition
- **Análisis Sintáctico**
- **Análisis Morfosintáctico (POS Tag)**
- Word Sense Disambiguation
- Speech Recognition
- Speech Generation
- **Sistemas de Diálogo**

Aplicaciones de PLN

Natural Language Processing

- Pos tagging
- Semantic Role labeling
- Named Entity Recognition

Natural Language Understanding

- Question Answering
- Text similarity

Natural Language Generation

- Sumarization
- Translation

- Sentiment Analysis
- Conversational Agents

Lingüística y PLN

Fonética y fonología	Morfología	Sintaxis
Semántica	Pragmática	Discurso

Figura 7. Conocimientos necesarios para tareas complejas de PLN.

Historia del Procesamiento de Lenguaje Natural

1940 1950	1957 1970	1970 1983	1983 1993	1993 1999	2000 2016	2016…
Automatas Modelos prob.	**Simbólico** Estocástico	**Estocástico** Lógica Comprensión Discurso	Estados finitos **Empirismo**	**Empirismo**	**Aprendizaje automático** Redes neuronales	**Deep Learning**

Historia del Procesamiento de Lenguaje Natural

Métodos formales

– Teoría de lenguajes formales:

- Un lenguaje **L** es un conjunto, finito o infinito de secuencias **s**
- Un vocabulario **V** es el conjunto de símbolos disponibles en el lenguaje
- Una secuencia **s** es una concatenación de símbolos de **V**
- Una gramática **G** es una definición formal de **L**: establece qué combinaciones de **V** forman secuencias **s** válidas en **L** y cuáles no

Métodos formales

- Teoría de autómatas:
  - Definición de lenguajes formales equivalente
  - Mediante máquinas abstractas
  - Formadas por un conjunto de estados **S** y una función de transición **T** que define transiciones de un estado a otro en función de los símbolos de una cadena de entrada

Métodos formales

Métodos formales

– ¿Puede el lenguaje natural formalizarse como un lenguaje formal **L**, mediante una gramática **G**, o como un autómata?

Métodos probabilísticos

- N-gramas
- Cálculo de probabilidades
- Cadenas de Markov (Hidden Markov Models o HMM)

Métodos probabilísticos

- N-gramas

  “por favor, por favor, no olvidemos que por este camino ya hemos pasado antes”

(por favor) 2

(favor ,) 2

(, por) 1

…

(por este) 1

(este camino) 1

…

Métodos probabilísticos

- Cálculo de probabilidades de N-gramas

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

Métodos probabilísticos

- #ocurrencias **por favor** en el corpus: 2
- #ocurrencias **por este** en el corpus: 1
- #ocurrencias **por** en el corpus: 3

$$P(favor|por) = \frac{C(porfavor)}{C(por)} = 0.66$$

$$P(este|por) = \frac{C(poreste)}{C(por)} = 0.33$$

Métodos probabilísticos

- Autómatas con probabilidades de transición

Machine Learning

- Machine Learning / Deep Learning
- Aprendizaje supervisado
- Proliferación de datos. Internet, redes sociales
- Soluciones de hardware: cloud, GPU, TPU
- Permiten modelar patrones demasiado complejos para formalizarlos explícitamente

Historia del PLN: aprendizaje automático

- Deep Learning
- Redes recurrentes: RNN, LSTM
- También convolucionales: CNN
- Transformadores: BERT
- Transfer Learning
- GAN aplicado a PLN

En resumen

- Aplicaciones de PLN
  - NLP (etiquetado),
  - NLU (question answering)
  - NLG (text generation)
- Lingüística y PLN
  - Morfología, sintaxis, semántica, pragmática, discurso
  - Fonética
- Historia del Procesamiento de Lenguaje Natural
  - Modelos formales
  - Modelos probabilísticos
  - Modelos basados en Machine Learning

En la próxima sesión

- ¿Cómo convertir el texto en una entrada válida para técnicas de IA?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 2 – El texto como dato

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Encuesta previa

- ¿Tienes instalado Jupyter? / ¿Sabes usar Google Colab?
- ¿Sabes qué es un corpus lingüístico?
- ¿Sabes la diferencia entre *lema* y *lexema*?

Problema del día

- ¿Cómo convertir el texto en una entrada válida para técnicas de IA?

Índice

- Principios conceptuales
- Morfología
- El pipeline de normalización
- Recursos
- Herramientas y librerías

Principios conceptuales

Principios conceptuales

- Idea principal
  - La estructura del lenguaje natural es demasiado compleja
- Solución
  - Simplificar los textos de forma sistemática

Principios conceptuales

- Simplificamos en base a tres principios:
  - Principio de composición
  - Aspecto secuencial
  - Perspectiva distribucional

Principio de composición

A nivel de oración:

El significado de una oración se construye como composición del significado de las palabras que contiene

Principio de composición

A nivel de oración:

El significado de una oración se construye como composición del significado de las palabras que contiene

A nivel de palabra:

La información de una palabra se construye como composición de la información que aportan sus morfemas

Aspecto secuencial

El orden en el que aparecen las palabras importa en la construcción del significado de una oración

“Un hombre pobre”

Vs

“Un pobre hombre”

“Al campo no he ido, he ido a la playa”

Vs

“A la playa no he ido, he ido al campo”

Perspectiva distribucional

El significado de una palabra en un texto depende de las palabras que la rodean

- Resulta útil en la solución de ambigüedades
  - … fue al **banco** a contratar una hipoteca.
  - … en el mar pude ver un **banco** de peces rojos.
  - … ellos se sentaron en un **banco** de madera.

Morfología

Principio composicional aplicado a palabras

Morfología de las palabras

- Morfemas :
  - **Lexema / morfema léxico**: raíz de la palabra con significado léxico
  - **Morfema gramatical**: significado gramatical - género, número, tiempo, etc.
    - **Morfema cero**: marca gramatical sin marca fonética (ej. mujer)
  - **Lema**: unidad autónoma con el significado léxico raíz de la palabra

Morfología de las palabras

- Ejemplos

Palabra	Lexema	Morfema gramatical	Morfema gramatical	Lema
Cantábais	Cant	ába	Is	Cantar
Casita	Cas	Ita	Morfema cero (SG)	Casa
Perros	Perr	O	S	Perro
Vino	Vino	Morfema cero (M)	Morfema cero (SG)	Vino

Morfología de las palabras

```
In [14]: import spacy
         import es_core_news_sm
```

```
In [15]: nlp = es_core_news_sm.load()
         doc = nlp("En un lugar de La Mancha")
```

```
In [17]: for token in doc:
             print("\"", token, "\": ", token.morph)
```

```
" En ":
" un ":  Definite=Ind|Gender=Masc|Number=Sing|PronType=Art
" lugar ":  Gender=Masc|Number=Sing
" de ":
" La ":  Definite=Def|Gender=Fem|Number=Sing|PronType=Art
" Mancha ":
```

El pipeline de normalización

Problema a resolver

- ¿Cómo construir una nube de palabras?

Pipeline

- Tokenización
- Borrado de StopWords
- NER
- Mayúsculas y minúsculas
- Lematización

Pipeline (detalle)

- Tokenización
  - “En un lugar de La Mancha, de cuyo nombre no quiero acordarme”
  - [“En”, “un”, “lugar”, “de”, “La”, “Mancha”, “, “, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]

```
In [4]: import spacy
        import es_core_news_sm
```

```
In [5]: nlp = es_core_news_sm.load()
        doc = nlp("En un lugar de La Mancha, de cuyo nombre no quiero acordarme")
```

Pipeline (detalle)

- Borrado de stopwords
  - [“En”, “un”, “lugar”, “de”, “La”, “Mancha”, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]
  - [“lugar”, “Mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]

```
for token in doc:
    if not(token.is_stop):
        print(token)
```

```
lugar
Mancha
,
cuyo
nombre
quiero
acordarme
```

Pipeline (detalle)

- NER
  - [“En”, “un”, “lugar”, “de”, **“La”, “Mancha”**, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]
  - [“En”, “un”, “lugar”, “de”, “**La_Mancha**”, “de”, “cuyo”, “nombre”, “no”, “quiero”, “acordarme”]

Pipeline (detalle)

- Mayúsculas / minúsculas
  - [“lugar”, “Mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]
  - [“lugar”, “mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]

```
for token in doc:
    if not(token.is_stop):
        print(token.lower_)
```

Pipeline (detalle)

- Lematización
  - [“lugar”, “mancha”, “cuyo”, “nombre , “quiero”, “acordarme”]
  - [“lugar”, “mancha”, “cuyo”, “nombre , “querer”, “acordar yo”]

```
for token in doc:
    if not(token.is_stop):
        print(token.lemma_)
```

Problema a resolver

- Nube de palabras: ¿qué efectos tendría añadir/quitar/cambiar el orden de estas etapas?

Recursos

Recursos lingüísticos disponibles

- Diccionarios
- Tesauros
- Lexicones

- Bases de datos lingüísticas
- Corpus

Diccionarios, Tesauros, Lexicones

- Diccionarios
  - RAE, Linguee, WordReference, TheFreeDictionary…
  - https://pypi.org/project/pyrae/
  - https://github.com/squat/drae
- Tesauros
  - TheFreeDictionary
- Lexicones

Bases de datos lingüísticas

- WordNet (https://wordnet.princeton.edu/)
- Spanish WordNet 3.0 (http://timm.ujaen.es/recursos/spanish-wordnet-3-0/)
- FrameNet (https://framenet.icsi.berkeley.edu/fndrupal/)

Corpus

- Un corpus es un conjunto de textos, que puede estar o no etiquetado, que se utiliza para tareas de procesamiento de leguaje natural
- Corpus de NLTK (https://www.nltk.org/nltk_data/)
- https://www.kaggle.com/datasets?tags=13204-NLP
- POS (https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1827)
- NER-C (https://www.kaggle.com/datasets/nltkdata/conll-corpora)
- MLDoc (https://github.com/facebookresearch/MLDoc)
- PAWS-X (https://github.com/google-research-datasets/paws/tree/master/pawsx)
- XNLI (https://github.com/facebookresearch/XNLI)

Herramientas y librerías

Herramientas y librerías

- NLTK
  - https://www.nltk.org

- Spacy
  - https://spacy.io

Para cerrar

Conceptos principales de esta sesión

- Normalizar es necesario para poder procesar
- El pipeline de normalización
  - Tokenizar -> quitar stopwords -> lematizar
- Recursos
  - Importante que sea posible acceder programáticamente
  - Corpus y Tesauros
- Herramientas y librerías
  - Spacy

En la próxima sesión

- ¿Cómo identificar la función de una palabra en una oración con técnicas estadísticas?
  - Y más concretamente:

¿Cómo construir tablas de transición y emisión?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 3 – Etiquetado Morfosintáctico *(POS Tagging)*

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Encuesta previa

- ¿Sabes qué es un corpus lingüístico?
- ¿Sabes qué es un Modelo de Markov?
- ¿Has abierto ya un notebook con algo de código para NLP?

Problema del día

¿Cómo construir tablas de transición y emisión?

En el día de hoy

- Definición de moforsintaxis
- Categorías morfosintácticas
- Hidden Markov Models para POS tagging
- Probabilidad de transición / Probabilidad de emisión

Morfosintaxis / POS Tagging

morfosintaxis

1. f. *Ling.* Parte de la gramática que integra la morfología y la sintaxis.

Morfosintaxis / POS Tagging

- La morfosintaxis es un subcampo de la gramática que tiene como objetivo el estudio de la formación de las palabras y las relaciones de éstas para conformar frases u oraciones coherentes, es decir, con sentido lógico. (https://conceptodefinicion.de/morfosintaxis/)

- Parte de la gramática que integra la morfología y la sintaxis para identificar las partes de una oración (apuntes de la asignatura)

Morfosintaxis / **POS Tagging**

- Part of Speech:
  - In grammar, a part of speech or part-of-speech (abbreviated as POS or PoS, also known as word class or grammatical category) is a **category of words** (or, more generally, of lexical items) that have similar grammatical properties. (Wikipedia)

Categorías Morfosintácticas

Las categorías gramaticales variables

- Sustantivos
- Pronombres
- Determinantes
- Adjetivos
- Verbos

Las categorías gramaticales invariables

- Adverbios
- Preposiciones
- Conjunciones
- Interjecciones

Categorías Morfosintácticas

Sustantivos

- Poseen género y número (morfología)
- Forman sintagmas nominales (sintaxis)
- Designan entidades de diferente naturaleza (semántica)

Categorías Morfosintácticas

Pronombres

- Sustituyen a los sustantivos y cumplen las mismas funciones
- Su significado concreto depende del **contexto** (p.ej. “ellos”)

Categorías Morfosintácticas

Determinantes

- Acompañan al sustantivo para precisar su significado
  - *El* coche
  - *Un* coche
  - *Ningún* coche
  - *Ese* coche

Categorías Morfosintácticas

Verbos

- Expresa una acción
- Es el núcleo del predicado
- Debe concordar en género y número con el sujeto
- Tiene variación de persona, número, tiempo, modo y aspecto

Etiquetado moforsintáctico

```
Bebo un vaso del vino tinto
```

Etiquetado moforsintáctico

Bebo un vaso del vino tinto

POS tagging informal

Bebo(verbo) un(determinante) vaso(sustantivo) de(preposición)
el(determinante) vino(sustantivo) tinto(adjetivo)

Etiquetado moforsintáctico

- Penn Treebank
  - Conjunto de etiquetas normalizadas
  - Diseñado para idioma inglés
  - https://gist.github.com/nlothian/9240750

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figura 1. Etiquetas para las categorías gramaticales en el Penn Treebank.
Fuente: Jurafsky y Martin, 2009.

Etiquetado moforsintáctico

```
Bebo un vaso del vino tinto
```

POS tagging informal

```
Bebo(verbo) un(determinante) vaso(sustantivo) de(preposición)
el(determinante) vino(sustantivo) tinto(adjetivo)
```

POS tagging con Penn Treebank

```
Bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ
```

Etiquetado moforsintáctico

- Alternativas al Penn Treebank
  - EAGLES (https://www.cs.upc.edu/~nlp/tools/parole-sp.html)

Talamé, L., Cardoso, A., & Amor, M. (2019). Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python. In *XX Simposio Argentino de Inteligencia Artificial (ASAI 2019)- JAIIO 48 (Salta)*.

Etiquetado moforsintáctico

- Demo con Spacy y NLTK

nltk-pos-tagging.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+ Código + Texto

```
#!pip install nltk
#import nltk
#nltk.download('punkt')
#nltk.download('averaged_perceptron_tagger')

from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
```

```
[ ] pos_tag(word_tokenize("I am loving this course"))
```

```
[ ] pos_tag(word_tokenize("Me está gustando mucho el curso"))
```

Hidden Markov Models para POS Tagging

- Modelo oculto de Markov (HMM)
  - Un modelo oculto de Markov es un modelo estadístico que se puede representar como una **máquina de estados finitos**, pero donde las **transiciones entre estados son probabilísticas** y no determinísticas

Hidden Markov Models para POS Tagging

- Propiedad de Markov
  - la distribución de probabilidad del valor futuro de una variable aleatoria depende únicamente de su valor presente, siendo independiente de la historia de dicha variable.

$$P(X_{n+1}|X_n)$$

- Proceso de Markov
  - Secuencia de observaciones en las que cada observación cumple la propiedad de Markov

Hidden Markov Models para POS Tagging

- Propiedad de Markov
  - la distribución de probabilidad del valor futuro de una variable aleatoria depende únicamente de su valor presente, siendo independiente de la historia de dicha variable.

$$P(X_{n+1}|X_n)$$

- Proceso de Markov
  - Secuencia de observaciones en las que cada observación cumple la propiedad de Markov

¿Es una oración un proceso de Markov?
Es decir, ¿se cumple la propiedad de Markov?

Hidden Markov Models para POS Tagging

- Modelo oculto de Markov
  - Modelo estadístico para determinar valores desconocidos (ocultos) a partir de valores observables en un proceso de Markov

```
Bebo/VBP un/DT vaso/NN de/IN el/DT vino/NN tinto/JJ
```

Parte Observable

Parte Oculta

Probabilidad de transición y de emisión

- Un HMM se compone de:
  - Conjunto de estados (POS tags)
  - Conjunto de observaciones (palabras | tags)
  - Matriz de **probabilidades de transición**
  - Matriz de **probabilidades de emisión**

Probabilidad de transición

- Las probabilidades de transición de etiqueta $P(ti|ti-1)$ representan la probabilidad de una etiqueta dada la etiqueta anterior.

- Matriz de **probabilidades de transición**

	PNN	MD	VB	JJ	NN	RB	DT
<S>	0.2767	0,0006	0,0031	0,0453	0,0449	0,0510	0.2026
PNN	0.3777	0,0110	0,0009	0,0084	0,0584	0,0090	0,0025
MD	0,0008	0,0002	0.7968	0,0005	0,0008	0.1698	0,0041
VB	0,0322	0,0005	0,0050	0,0837	0,0615	0,0514	0.2231
JJ	0,0366	0,0004	0,0001	0,0733	0.4509	0,0036	0,0036
NN	0,0096	0,0176	0,0014	0,0086	0.1216	0,0177	0,0068
RB	0,0068	0,0102	0.1011	0.1012	0,0120	0,0728	0,0479
DT	0.1147	0,0021	0,0002	0.2157	0.4744	0,0102	0,0017

Figura 4. Conjunto A de probabilidades de transición $P(t_i|t_{i-1})$ calculadas a partir del corpus WSJ. Fuente: Jurafsky y Martin, 2009.

Probabilidad de transición

- Las probabilidades de transición de etiqueta $P(ti|ti-1)$ representan la probabilidad de una etiqueta dada la etiqueta anterior.

- Matriz de **probabilidades de transición**

	PNN	MD	VB	JJ	NN	RB	DT
<S>	0.2767	0,0006	0,0031	0,0453	0,0449	0,0510	0.2026
PNN	0.3777	0,0110	0,0009	0,0084	0,0584	0,0090	0,0025
MD	0,0008	0,0002	0.7968	0,0005	0,0008	0.1698	0,0041
VB	0,0322	0,0005	0,0050	0,0837	0,0615	0,0514	0.2231
JJ	0,0366	0,0004	0,0001	0,0733	0.4509	0,0036	0,0036
NN	0,0096	0,0176	0,0014	0,0086	0.1216	0,0177	0,0068
RB	0,0068	0,0102	0.1011	0.1012	0,0120	0,0728	0,0479
DT	0.1147	0,0021	0,0002	0.2157	0.4744	0,0102	0,0017

Figura 4. Conjunto A de probabilidades de transición $P(t_i|t_{i-1})$ calculadas a partir del corpus WSJ. Fuente: Jurafsky y Martin, 2009.

Se calcula a partir de un **corpus de trabajo**, y por tanto los valores dependen de dicho corpus

En la matriz aparecen los posibles estados, y es completamente **independiente de la oración** observada

Probabilidad de emisión

- Es probabilidad de que, dada una etiqueta, esta se asocie con una palabra concreta

$$P(w_i|t_i)$$

- Matriz de **probabilidades de emisión**

	Janet	will	back	the	bill
PNN	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figura 5. Conjunto B de probabilidades de observación calculadas a partir del corpus WSJ.
Fuente: Jurafsky y Martin, 2009.

Probabilidad de emisión

- Es probabilidad de que, dada una etiqueta, esta se asocie con una palabra concreta

$$P(w_i|t_i)$$

- Matriz de **probabilidades de emisión**

	Janet	will	back	the	bill
PNN	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figura 5. Conjunto B de probabilidades de observación calculadas a partir del corpus WSJ.
Fuente: Jurafsky y Martin, 2009.

Se calcula a partir de un **corpus de trabajo**, y por tanto los valores dependen de dicho corpus

En la matriz aparecen los posibles estados y **las palabras observadas**.

En resumen

- Etiquetado morfosintáctico o POS tagging
  - Asignar a cada palabra su categoría gramatical, de acuerdo al contexto concreto de la oración observada
- Hidden Markov Models para POS tagging
  - Modelo probabilístico en el que vamos a basar los algoritmos POS tagging
  - Requiere asumir cosas que no son del todo ciertas
- Probabilidad de transición / Probabilidad de emisión
  - Transición: dado un estado, probabilidad de cambiar a otro
  - Emisión: dado un estado observado, probabilidad de una observación concreta
  - Se calculan para un corpus dado

En la próxima sesión

¿Cómo identificar la función de una palabra en una oración con técnicas estadísticas?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python

Lorena Talamé, Alejandra Cardoso, Matías Amor[1]

[1] Universidad Católica de Salta,
Campo Castañares s/n, 4400, Salta

{mltalame, acardoso, matiasamor}@ucasal.edu.ar

Resumen. El procesamiento de lenguaje natural es un área central en cualquier aplicación de análisis de datos textuales. Muchas de estas aplicaciones se realizan a partir de textos publicados en Internet y muy frecuentemente, a partir de textos de redes sociales. Existen diversas herramientas informáticas de código libre que facilitan el procesamiento de lenguaje natural de diversas lenguas, pero especialmente para el idioma inglés. En este artículo se seleccionaron cinco módulos o librerías para el lenguaje Python, con el objetivo de comparar algunas funciones básicas del procesamiento de textos, aplicadas a textos cortos en español extraídos de una red social. La tokenización y segmentación de oraciones fueron perfectamente realizadas por dichas herramientas. Se encontraron diferencias en etiquetado gramatical pero los resultados se consideran aceptables.

Keywords: procesamiento de lenguaje natural, etiquetado gramatical, Python

1. Introducción

Con el uso de internet y redes sociales, compartimos información (multimedial o textual) que puede resultar valiosa en diversos ámbitos. Muchas personas utilizan las redes sociales para compartir su opinión referida a un evento, a un producto o a una situación en particular. Por lo tanto, resulta interesante poder obtenerlas y analizarlas automáticamente.

Por otro lado, la proporción de la población hispanohablante va en aumento. El español, en el mundo y como lengua materna, ocupa el segundo lugar, luego del chino mandarín [1]. Si nos centramos en los usuarios de Internet, las lenguas que más se emplean son el inglés, el chino y, en el tercer puesto, el español. Sin embargo, si se contempla que el chino, en general, lo usan los nativos, el español se situaría en segundo lugar tras el inglés.

El procesamiento de lenguaje natural (abreviado PLN o en sus siglas en inglés NLP, natural language processing) estudia la interacción entre una computadora y una persona facilitando la comunicación en lenguas naturales o humanas, como el idioma inglés o el español. En la actualidad, es un área de investigación y producción científica muy activa con numerosas aplicaciones.

Existen diversas herramientas informáticas que facilitan el procesamiento de texto en múltiples idiomas, especialmente para el idioma inglés. En este artículo se analizan algunas herramientas de código libre para el lenguaje Python, poniendo el acento especialmente en el análisis morfológico (determinar la forma o tipo gramatical de cada palabra o token en una oración) de textos cortos en lenguaje español, extraídos de la red social Twitter. El trabajo descrito en este artículo forma parte de un proyecto de mayor alcance que aún está en desarrollo.

2. Procesamiento de lenguaje

El procesamiento de lenguaje natural es un área de la inteligencia artificial que estudia la comunicación entre las personas y las computadoras utilizando lenguas naturales. Los lenguajes naturales son aquellos que utilizan los humanos para comunicarse ya sea en forma escrita u oral. Algunas aplicaciones típicas del PLN son la búsqueda de respuestas, corrección ortográfica, reconocimiento de voz, generación automática de resúmenes, traducción automática y análisis de sentimientos, entre otras. El proceso del PLN, generalmente, contempla cuatro tipos de análisis sobre la entrada de un texto en lenguaje natural (Fig. 1).

Fig. 1. Proceso PLN

A continuación, se describen los análisis indicados en la figura anterior:

- *análisis morfológico*: consiste en el análisis de las oraciones para identificar unidades léxicas o tokens (unidades del lenguaje con significación propia como palabras o signos de puntuación) y tipos de palabras (sustantivos, verbos, etc.). Una misma palabra puede adoptar diferentes roles en función del contexto en el que aparece, ocasionando problemas de ambigüedad. En el análisis morfológico se determina la forma, clase o categoría gramatical de cada palabra, es decir, se asigna a cada palabra de la oración, el rol que cumple dentro de ella. Esto se conoce como etiquetado gramatical o *POS* (Part-of-speech) *tagging*, el cual debe resolver las ambigüedades que presentan algunas palabras según el contexto donde se encuentren.

- *análisis sintáctico*: el objetivo de este análisis es determinar las relaciones de dependencia estructural entre las palabras de la oración estudiada. Esta dependencia y el orden de las palabras contribuyen a su significado. En la mayoría de las lenguas, este significado se transmite por la sintaxis. Por ejemplo, las dos oraciones: "*A ella le gusta Miguel de Cervantes*" y "*A Miguel de Cervantes le gusta ella*" difieren sólo en términos de sintaxis, pero transmiten significados diferentes. En este nivel se realiza

la desambiguación sintáctica de palabras que pueden funcionar como múltiples partes del discurso. La automatización de esta tarea se dificulta por la ambigüedad del lenguaje [2].

- *análisis semántico*: incluye la desambiguación de palabras con múltiples sentidos. Si se requiriera información del resto de la oración para esta desambiguación, se realizaría en este nivel. Se puede implementar una amplia gama de métodos para lograr la desambiguación, algunos de los cuales requieren información sobre la frecuencia con que ocurre cada sentido de las palabras, en un corpus particular de interés, y otros utilizan el conocimiento pragmático del dominio del documento en cuestión.

- *análisis pragmático*: es uno de los más complejos junto con el semántico. En muchas ocasiones las oraciones, y las palabras que las componen, no pueden ser interpretadas de manera literal ya que su significado depende en gran medida del contexto en el que se utilicen. Entonces, el análisis pragmático, a diferencia del sintáctico y semántico, va un paso más allá de la estructura del lenguaje, ya que es necesario entender el contexto donde las oraciones son emitidas para otorgarles significado.

Cabe mencionar que no siempre se aplican todos los niveles de análisis a cualquier tarea de PLN, depende del objetivo de la aplicación. En este trabajo nos enfocamos en el análisis morfológico.

Generalmente, el PLN necesita algunas acciones de “normalización” de los textos para poder “estructurarlos” y así analizarlos de una manera más fácil. A continuación, se definen algunos de estos procesos.

Tokenización

La tokenización es el proceso de dividir un documento de texto en sus distintos componentes, descartando los espacios en blancos y saltos de línea. Un token es una cadena de caracteres (palabra o un signo de puntuación) con algún significado en el contexto de un texto. Por ejemplo, el texto “El niño corre” tiene tres tokens: {'El', 'niño', 'corre'}.

Aunque la tokenización puede resultar trivial, no lo es si se considera que existen caracteres que algunas veces pueden ser parte de un token y otras veces pueden resultar ser separadores de tokens. Por ejemplo, el carácter punto puede ser parte de una abreviatura o sigla, pero también puede resultar ser separador de oraciones.

Segmentación

Consiste en separar el texto en fragmentos que puedan tratarse de forma independiente. La forma más usual de segmentación es dividir el texto en párrafos u oraciones.

Etiquetado gramatical

El etiquetado gramatical o *POS (part-of-speech) Tagging* consiste en la asignación de una etiqueta o categoría gramatical (sustantivo, verbo, adjetivo, etc.) a cada uno de los tokens de una oración según el rol que cumpla dentro de la misma. Por ejemplo,

para la oración "*El niño corre por la plaza*" el resultado del etiquetado POS se observa en la segunda fila de la Tabla 1.

Tabla 1. Ejemplo etiquetado POS

Token	**El**	**niño**	**corre**	**por**	**la**	**plaza**
Etiquetado POS	Articulo masculino singular	Nombre común masculino singular	Verbo indicativo presente 3ra. persona del singular	Preposición	Articulo femenino singular	Nombre común femenino singular

Los algoritmos de POS-tagging deben resolver las ambigüedades que presentan algunas palabras según el contexto donde se encuentren. Por ejemplo, en la oración "*El niño ayuda a su hermana*", la palabra "ayuda" se define como verbo; sin embargo, en la oración "*El niño necesita ayuda*" la misma palabra se establece como sustantivo.

Las categorías gramaticales resultan de gran utilidad por la información que proporcionan acerca de una palabra, sus palabras continuas y la forma de interpretarlas. Saber si una palabra es un sustantivo o un verbo dice mucho sobre las palabras junto a ellas y las formas de interpretarlas. Sirve además para encontrar entidades nombradas en textos y se aplica también a otras tareas de extracción de información.

Reconocimiento de entidades nombradas

El reconocimiento de entidades con nombres o entidades nombradas (*NER, Named Entity Recognition*) se ocupa de la extracción de unidades de información predefinidas, como nombres de personas, fechas, lugares, organizaciones, etc.

3. Herramientas informáticas para PLN

Existe una amplia variedad de herramientas informáticas para el PLN en diversos idiomas. En este trabajo se seleccionaron algunas Open Source que se pueden utilizar en programas Python[1]. La comunidad científica utiliza en mayor medida el lenguaje interpretado Python por lo que sólo nos centraremos en aquellas bibliotecas que se puedan utilizar en este lenguaje de programación. Cabe mencionar que, si bien no todas son nativas para este lenguaje, como Stanford NLP, existen plugins o APIs que permiten enlazarlas. En el contexto de este trabajo se utilizarán los términos "herramientas", "librerías", "bibliotecas" y "paquetes" como sinónimos.

[1] https://www.python.org

3.1. NLTK

NLTK[2] (Natural Language Toolkit) es una plataforma para la creación de programas Python para análisis de texto. Como fue creado con fines educativos en el año 2001, permite la realización de proyectos con distintos propósitos y alcance variado. Provee demostraciones paso a paso de distintos algoritmos [3]. Integra más de 50 corpus y un conjunto de librerías para segmentación, tokenización, etiquetado del habla, análisis sintáctico y semántico, entre otros. Para el español muchas de estas opciones no están disponibles.

Su instalación en Python es sencilla debido a que es un paquete que se encuentra en el repositorio del lenguaje. Con una simple instrucción, se instala lo necesario para utilizarlo.

El etiquetador POS de NLTK originalmente fue creado para el idioma inglés. Sin embargo, se lo puede extender al español entrenando el etiquetador con algún corpus que incorpora NLTK en esta lengua o usando un etiquetador externo como el de Stanford. Para este trabajo se seleccionó la primera opción con el corpus cess_esp [4] que tiene 500.000 palabras y 610 archivos.

El etiquetado POS que retorna NLTK se corresponde con el formato EAGLES[3]. Las etiquetas EAGLES codifican todas las características morfológicas existentes para la mayoría de los idiomas europeos, incluido el español. Estas etiquetas consisten en un conjunto de caracteres de longitud variable donde cada uno corresponde a una característica morfológica. El primer carácter en la etiqueta es siempre la categoría POS que, a su vez, determina la longitud y la interpretación de los elementos siguientes [5].

3.2. Freeling

Freeling[4] es una librería de código abierto para el análisis de texto (tokenización, análisis morfológico, detección de entidades nombradas, etiquetado POS, etc.) para una variedad de idiomas. Desde su primera versión, tiene soporte para el español. Fue desarrollado por el Centro de Tecnologías y Aplicaciones del Lenguaje y el Habla (TALP) de la Universidad Politécnica de Catalunya. Se puede utilizar y ampliar los recursos lingüísticos por defecto (diccionarios, lexicones, gramáticas, etc.) adaptándolos a dominios específicos.

Su código fue escrito en C++ bajo una arquitectura cliente – servidor. Existen diversas APIs que permiten utilizar Freeling en otros lenguajes, incluido Python. Su instalación no es sencilla, sobre todo en entornos Windows. Sin embargo, subsanado este inconveniente, al importar la API desde el programa Python se puede invocar a todas las funciones implementadas.

El analizador morfológico de Freeling realiza el etiquetado POS usando las etiquetas EAGLES.

---

[2] https://www.nltk.org/
[3] http://blade10.cs.upc.edu/freeling-old/doc/tagsets/tagset-es.html
[4] http://nlp.lsi.upc.edu/freeling/node/1

3.3. Pattern.es

Pattern.es[5] es una biblioteca para Python que contiene herramientas para la conjugación de verbos, la singularización o la pluralización de sustantivos, la división de chunks y permite realizar el etiquetado POS para español. En la versión para el idioma inglés adiciona una interfaz para WordNet. Su instalación y uso son muy sencillos.

Para el etiquetado gramatical utiliza Penn TreeBank Tags [6]. Penn TreeBank está basado en el corpus Brown, pionero en etiquetado POS para inglés. A pesar de haberse creado para el inglés, Pattern lo usa también para el español. Cada etiqueta está formada con dos o tres caracteres que indican la función que cumple cada palabra en una oración.

3.4. SPACY

SpaCy[6] es una biblioteca para el procesamiento avanzado de lenguaje natural en Python. Incluye modelos estadísticos pre-entrenados y vectores de palabras, admite tokenización para más de 45 idiomas. Provee etiquetado, análisis y reconocimiento de entidades nombradas y una fácil integración de aprendizaje profundo. Es muy sencilla de utilizar, basta con incorporar desde un programa Python el modelo para el español.

Devuelve un etiquetado POS completo, no sólo indica la función de la palabra en la oración, sino otros datos tales como tiempo verbal, persona, número, modo, género, entre otros. La tokenizacion y el etiquetado gramatical se basan en el corpus OntoNotes5[7] que sigue la sintaxis de Penn TreeBank.

3.5. Stanford NLP

Stanford NLP es un framework diseñado y desarrollado para Java que provee soluciones para la mayoría de las tareas más comunes en el procesamiento de lenguaje natural que van desde la tokenización hasta el reconocimiento del discurso [7]. No todas las funciones están implementadas para el español.

Existen APIs para otros lenguajes de programación, entre ellos, Python[8]. NLTK incluye algunas de sus funciones, aunque si se desea hacer un análisis más profundo, siempre se debe correr el servidor de Java en segundo plano y prescindir de NLTK.

Permite realizar etiquetado POS con el formato EAGLES y reconoce entidades para el análisis morfológico. Para el análisis sintáctico, identifica grupos lingüísticos o chunks y se puede obtener el árbol de dependencia.

Su instalación no es tan sencilla como otras de las herramientas analizadas por el hecho de tener que instalar el servidor Java y definir variables de entorno.

---

[5] https://www.clips.uantwerpen.be/pages/pattern-es
[6] https://spacy.io/
[7] https://catalog.ldc.upenn.edu/LDC2013T19
[8] https://stanfordnlp.github.io/stanfordnlp/

3.6. Comparación

De cada herramienta se tuvieron en cuenta algunos aspectos como la facilidad en su instalación, sobre todo para una persona con conocimientos básicos o nulos en programación; la necesidad de utilizar una interfaz de comunicación; la presencia de funciones específicas para segmentación y tokenización; el tipo de etiquetado POS que realizan y la implementación de lematizador para el lenguaje español.

A modo de resumen se presenta un cuadro comparativo de algunas funciones básicas de procesamiento de texto y las herramientas mencionadas que fueron objeto de evaluación (Tabla 2).

Tabla 2. Comparación de herramientas

Herramienta	Código	Segmentación y tokenización	Etiquetado POS	Lematización
NLTK	Nativo Python	Si	Eagles	No
Freeling	API	Si	Eagles	Si
Pattern.es	Nativo Python	Si	Penn TreeBank	Si
Spacy	Nativo Python	Si	Penn TreeBank	Si
Stanford NLP	API	Si	Eagles	Si

En la segunda columna de la Tabla 2 se muestra la relación de las herramientas con el lenguaje Python. NLTK, Pattern.es y Spacy se pueden instalar muy fácilmente de los repositorios de Python a través de la línea de comando. No ocurre lo mismo con Stanford NLP y Freeling. Ambos paquetes no son nativos de Python; fueron escritos en Java y C++, respectivamente. La instalación de Stanford es más sencilla que la de Freeling aunque ambos necesitan sendos servidores que se estén ejecutando para funcionar. Sin embargo, si no se requiere de todo el potencial de ambas herramientas, existe otra opción para su uso: Stanford se puede utilizar a través de NLTK y Freeling provee un ejecutable que realiza la segmentación, la tokenización, el etiquetado POS y la lematización recibiendo un archivo como entrada.

Todas las herramientas poseen funciones para segmentación y tokenización. NLTK, Freeling y Stanford NLP realizan el etiquetado POS con etiquetas Eagles mientras que Pattern.es y Spacy lo hacen con etiquetas de Penn TreeBank.

Salvo NLTK, las demás herramientas, implementan un lematizador para el idioma español. El lematizador disponible en NLTK, WordNet, está disponible para el idioma inglés.

4. Corpus

Para poder evaluar las diferentes alternativas mencionadas para el procesamiento del lenguaje natural, se diseñó un corpus de textos cortos obtenidos de la red social Twitter (llamados *tweets*), por lo cual los textos que serán objeto de estudio son subjetivos o coloquiales. Se capturaron tweets utilizando la API[9] de Twitter y a partir de una serie de hashtags sobre temas de actualidad de Argentina durante Octubre del año 2018. Los mensajes se almacenaron en una base de datos NoSQL. Se optó por trabajar con mensajes en lenguaje español, generados en nuestro país (según identificador de geolocalización) y se evitó mensajes con errores ortográficos. Si bien se recopilaron muchos tweets durante este mes, se descartaron una gran parte de ellos por contener solo imágenes, íconos o textos con poca información para el análisis posterior. Finalmente se seleccionaron al azar 100 mensajes, algunos de los ellos compuestos por más de una oración.

Como el objetivo de esta experimentación fue comparar las devoluciones de cada herramienta de PLN, en primer lugar, fue necesario realizar una serie de acciones para eliminar íconos, emojis, hashtags y links, los cuales son muy usuales en estos tipos de textos.

4.1. Preparación de documentos

Muchos comentarios en las redes sociales, no suelen tener en cuenta las reglas ortográficas, utilizan palabras del lunfardo o abreviaturas y muchas veces acentúan sus emociones con el uso de *emojis* (pequeñas imágenes que representan emociones, comidas, lugares, etc.), palabras completas en mayúsculas o signos de exclamación. Por esta razón, se realizaron una serie de acciones a modo de limpieza de los textos (Fig. 2) previo al análisis, ya que las herramientas analizadas sólo reconocen palabras válidas del idioma español.

Fig. 2. Preparación de documentos

Como primer paso, teniendo en cuenta que el objetivo del trabajo es comparar las herramientas a nivel textual, se eliminaron los emojis, url's y hashtags presentes en los tweets. Para esto se escribió un programa Python y se utilizaron expresiones regulares y la librería *emoji* que detecta símbolos Unicode.

La repetición de letras es usual para intensificar la emoción en un texto. Por ejemplo, "*Es taaaaaan pero tan difícil que yo me aleje de alguien que quería, pero si lo hice es*

[9] https://developer.twitter.com/

porque de verdad ya no aguantaba más.". En este caso, la palabra "*taaaaaan*" demuestra el acento que quiso poner el usuario, pero no es una palabra válida en español. Por esto, se descartaron las letras repetidas e innecesarias para tener una palabra reconocida en la lengua española [8]. En el ejemplo, se reemplazaron las seis letras "*a*" por una sola obteniendo como resultado "*tan*" que sí es correcta en esta lengua.

La última tarea realizada fue reemplazar aquellas abreviaciones que comúnmente se utilizan por la palabra completa. Por ejemplo, se reemplazó el "*x*" por "*por*" y "*xq*" por "*porque*", entre otras.

5. Experimentos

Para el estudio comparativo se evaluaron las funciones de segmentación de oraciones, tokenización y etiquetado POS en cada herramienta. Con el objeto de medir la capacidad de cada una y de tener un punto de comparación, se aplicaron, manualmente, éstas formas de procesamiento al corpus. Para facilitar las comparaciones posteriores se equipararon las etiquetas EAGLES y Penn TreeBank, definiendo nuevas etiquetas (segunda columna de Tabla 3) con las cuales se etiquetó el corpus. Las columnas siguientes de la tabla, contienen las etiquetas utilizadas por los analizadores (EAGLES y Penn TreeBank) y ejemplos. Nótese que de EAGLES solo se utilizaron los primeros caracteres que son los que contienen la información principal de cada etiqueta.

Tabla 3. Equivalencia de etiquetas

Descripción	Etiqueta	EAGLES	TreeBank	Ejemplo
Conjunción	CONJ	CC	CC	y, o
Determinante	DET	DA	DT	la, el
Adjetivo	ADJ	AQ	JJ	alegre, mala
Verbo	VERBO	VM, VA, VS	VB, VBZ, VBP	estoy, suman
Sustantivo	SUST	NC	NN, NNS	circo, noche
Nombre Propio	NPRO	NP	NP, NPS	Argentina
Adverbio	ADV	RG	ADP	siempre, pronto
Interjección	INTJ	I	IN	ah, ay
Signos Puntuación	PUNT	FAT, FP, FC	O	. ; !

5.1. Segmentación

El corpus está compuesto de 136 oraciones (algunos tweets contienen más de una oración) muchas de las cuales no estaban correctamente separadas dentro de cada tweet, es decir, luego de punto al final de oraciones no se encontró un espacio antes del comienzo de la oración siguiente, por ejemplo: "*Nosotros vivíamos en el campo.Nos relacionábamos con familias amigas*". Otro problema encontrado en el corpus es que muchos tweets después de un punto no comenzaban con mayúsculas. Un ejemplo de esto es "*Mi papá se jubila e invita a los vecinos a un costillar. tremenda felicidad*". Además, a otros comentarios les faltaba el punto para diferenciar una sentencia de otra. Para algunos textos, NLTK detectó más oraciones de las que realmente tienen. Estos son los casos que contienen más de un signo de puntuación consecutivo. Por ejemplo: *"¡¡¡Vacaciones!!! Lo bueno es que desperté y ya sabía que era domingo.*" Detectó tres oraciones: *"¡¡¡Vacaciones!!*", *"!*" y *"Lo bueno es que desperté y ya sabía que era domingo.* "

En primer lugar, se evaluó la capacidad de las herramientas para la segmentación. A pesar de los inconvenientes mencionados, todas las herramientas lograron distinguir las oraciones en un porcentaje aceptable, Stanford NLP logró el mejor resultado segmentando el 98,53% de las oraciones. En la segunda columna de la Tabla 4 se observa, la cantidad de oraciones detectadas por cada analizador y en la última columna los porcentajes correspondientes.

Tabla 4. Resultados segmentación

Oraciones en corpus	**136**	
Stanford NLP	134	98.53%
NLTK	129	94.85%
SPACY	124	91.18%
Freeling	123	90.44%
Pattern.es	128	94.17%

Para todo el análisis que continúa se solucionaron los problemas de separación de oraciones. De esta manera, dos herramientas lograron detectar la totalidad de sentencias: Stanford NLP y NLTK. Pattern sólo no encontró una. Freeling aumentó su rendimiento detectando el 95,58% de las oraciones. Spacy sólo encontró cuatro sentencias más que antes logrando un 94,12% de eficiencia. En la Tabla 5 se reflejan estos resultados.

5.2. Tokenización

Nltk, Spacy y Pattern no tokenizaron correctamente aquellas palabras precedidas por un signo de exclamación o interrogación. Por ejemplo, en la oración *"¡No creo en el circo de la información!"*, se consideró "*¡No*" como un único token cuando lo correcto son dos tokens: "*¡*" y "*No*".

En el corpus se identificaron 1481 tokens (palabras, signos de puntuación, números y fechas), de los cuales más del 90% fueron correctamente identificados por todas las herramientas.

Tabla 5. Resultados de segmentación con textos corregidos

Oraciones en corpus	**136**	
Stanford NLP	136	100%
NLTK	136	100%
SPACY	128	94.12%
Freeling	130	95.58%
Pattern.es	135	99.26%

5.3. Etiquetado POS

Las herramientas utilizan distintas nomenclaturas para el etiquetado POS pero, como se mencionó antes, se realizó una equiparación para permitir la comparación. En la Tabla 6 se observa el resultado de la evaluación de la oración *"¡No creo en el circo de la información!"*. En la segunda fila se observa el etiquetado realizado manualmente y en las filas posteriores las etiquetas devueltas por cada analizador (equiparadas según se describe en la Tabla 3). Obsérvese que "DESC" representa la etiqueta "None" que devuelve Nltk al no poder clasificar.

Tabla 6. Ejemplo etiquetado POS

Tweet	***¡***	***No***	***creo***	***en***	***el***	***circo***	***de***	***la***	***información***	***!***
	PUNT	**ADV**	**VERBO**	**PREP**	**DET**	**SUST**	**PREP**	**DET**	**SUST**	**PUNT**
Nltk	DESC		VERBO	PREP	DET	SUST	PREP	DET	SUST	PUNT
Freeling	PUNT	ADV	VERBO	PREP	DET	SUST	PREP	DET	SUST	PUNT
Pattern	NPRO		VERBO	INTJ	DET	SUST	INTJ	DET	SUST	PUNT
Spacy	SUST		VERBO	ADV	DET	SUST	ADV	DET	SUST	PUNT
Stanford	PUNT	ADV	VERBO	PREP	DET	SUST	PREP	DET	SUST	PUNT

A continuación, se describen algunas particularidades detectadas de cada librería.

NLTK

Una de las principales dificultades de esta herramienta es que no logra separar palabras precedidas o seguida de signos de puntuación (punto, exclamación o interrogación). Muchos de los casos analizados fueron etiquetados como "None", perdiendo así información muy útil sobre el etiquetado. Algunos ejemplos: "*cambiare!",* "*cariño?*".

FREELING

Si bien en este estudio no se realizó el reconocimiento de entidades nombradas es importante mencionar que Freeling las detecta de manera más acertada que el resto de las herramientas. Por ejemplo, en la oración "*La mejor noche con mis amigos en San Salvador de Jujuy*", Freeling indica que [San_Salvador_de_Jujuy] es un nombre propio (Tabla 7).

Tabla 7. Ejemplo 1

Tweet	***La***	***mejor***	***noche***	***con***	***amigos***	***en***	***San***	***Salvador***	***de***	***Jujuy***
Nltk	DT	ADJ	SUST	PREP	SUST	PREP	DESC	NPROP	PREP	DESC
Freeling	DT	ADJ	SUST	PREP	SUST	PREP	NPROP			
Pattern	DT	ADJ	SUST	PREP	SUST	INTJ	ADJ	NPROP	INTJ	NPROP
Spacy	DT	ADJ	SUST	ADV	SUST	ADV	NPROP	NPROP	ADV	NPROP
Stanford	DT	ADJ	SUST	PREP	SUST	PREP	NPROP	NPROP	PREP	NPROP

Solo algunas palabras (27) que comienzan con mayúsculas las clasifica como nombre propio cuando en realidad no lo son, por ejemplo "*Te Amo*". Este es un problema que encontramos en este tipo de textos, donde no siempre se respetan las reglas ortográficas. Freeling también identifica adverbios compuestos por más de una palabra, por ejemplo: "*por favor*" lo clasifica como adverbio (en este caso, adverbio de modo) mientras que las otras herramientas indican que son preposición y sustantivo (ejemplo en Tabla 8).

Tabla 8. Ejemplo 2

Tweet	***Por***	***favor***	***,***	***no***	***confundas***	***a***	***la***	***gente***
Nltk	PREP	SUST	PUNT	ADV	DESC	PREP	DT	SUST
Freeling	ADV		PUNT	ADV	VERBO	PREP	DT	SUST
Pattern	INTJ	ADV	PUNT	ADV	SUST	INTJ	DT	SUST
Spacy	ADV	SUST	PUNT	ADV	ADJ	ADV	DT	SUST
Stanford	PREP	SUST	PUNT	ADV	VERBO	PREP	DT	SUST

PATTERN

En el corpus existen cuatro menciones a días de la semana (ejemplo "*Primera vez que no quiero que llegue el viernes*"). Pattern los clasifica como nombres comunes o sustantivos, sin embargo, las otras herramientas les asignan la etiqueta "fecha".

Al igual que NLTK, no detecta los signos de exclamación e interrogación de apertura. A las preposiciones las identifica como interjecciones.

SPACY

A muchas de las preposiciones las clasifica como adverbios y las fechas como sustantivo. Sin embargo, si se incluyen otros argumentos al análisis, se observa que las identifica correctamente, es decir devuelve adtype=prep y advtype= time, respectivamente.

STANFORD

Tanto Stanford cono Freeling, a las contracciones las dividen en dos palabras. Por ejemplo, la contracción "del" la separa en "de" y en "el" asignándoles las etiquetas preposición y determinante, respectivamente.

De los diez números existentes en el corpus, solo identifica nueve. Las palabras "sábado" y "viernes" se clasifican como sustantivo, a diferencia de Freeling que les asigna la etiqueta "fecha".

6. Resultados

Freeling etiquetó correcta y completamente los tweets, en el 28% de los casos y Stanford en el 19%, el resto de las herramientas no superaron el 6%, y esto se debe, a que en muchos textos Spacy, Nltk y Pattern no diferencian los tokens que comienzan con signos de puntuación. Se observa que modificando los textos para una correcta tokenización, los porcentajes de estas últimas herramientas mejoran, pero no superan

los valores obtenidos por Freeling y Stanford. El resto de los textos tuvieron coincidencia en gran parte del etiquetado, así, todas las herramientas coincidieron con el etiquetado manual entre el 64% y 75% (Tabla 9).

Tabla 9. Resultados evaluación

	Nltk	**Freeling**	**Pattern**	**Spacy**	**Santford**
Total	3%	28%	4%	6%	19%
Parcial	72%	64%	64%	75%	72%

7. Conclusiones

El procesamiento de lenguaje natural es un área central en cualquier aplicación de análisis de datos textuales. Muchas de estas aplicaciones se realizan a partir de textos publicados en Internet y muy frecuentemente, a partir de textos de redes sociales. Por otro lado, la variedad de herramientas Open Source para procesamiento de textos es amplia, lo cual facilita cualquier tipo de investigación en este campo. En este artículo se utilizaron cinco herramientas o librerías para evaluar sus resultados sobre un corpus de textos cortos, en español y con términos coloquiales.

A diferencia de otros trabajos que realizaron comparaciones de herramientas de PLN, para la creación y explotación de corpus [9] se utilizaron textos escolares o fuentes formales (por ejemplo en [10] se construyó un corpus de sentencias), el corpus de textos de este trabajo fue obtenido de una red social. Los comentarios en redes sociales suelen contener imágenes, links, íconos, emojis, y muchas veces contenido textual incorrecto desde el punto de vista ortográfico. Por esta característica fue necesaria una etapa de preparación previa al análisis de las herramientas.

Se compararon tres funciones básicas del PLN: segmentación de oraciones, tokenización y etiquetado gramatical. Respecto a los dos primeros no se observaron mayores diferencias. Considerando el etiquetado completo y correcto de cada oración, Freeling tuvo la mayor cantidad de aciertos.

Desde el punto de vista de la sencillez de instalación, las herramientas nativas de Python permiten utilizar repositorios lo que facilita su uso a personas no expertas en programación. Tanto Stanford como Freeling, necesitan conocimiento más específico para utilizar sendos servidores.

El resultado de esta comparación será utilizado en otro proyecto que está desarrollando este equipo de investigación. Aunque las herramientas analizadas proveen más opciones para el análisis de textos, nuestro objetivo consistió en evaluar sólo aquellas que serán utilizadas en el proyecto principal.

Referencias

[1] Instituto Cervantes, «El español: una lengua viva. Informe 2018,» 2018. [En línea]. Available: https://cvc.cervantes.es/lengua/espanol_lengua_viva/pdf/espanol_lengua_viva_2018.pdf. [Último acceso: 02 2019].

[2] S. N. Galia Haro y A. Gelbukh, Investigaciones en análisis sintáctico para el español, México: Instituto Politécnico Nacional, Dirección de Publicaciones, 2007.

[3] S. Bird y E. Loper, «NLTK: The Natural Language Toolkit,» *ACL Association for Computacional Linguistics,* 2002.

[4] M. Martí, M. Taulé, L. Márquez y M. Bertran, «Cess-ece: corpus anotados del español y catalán.,» *Arena Romanistica. A New Nordic Journal of Romance Studies,* vol. 1, 2007.

[5] Research Center on Language and Speech Technologies and Applications, «Freeling 4.1,» [En línea]. Available: https://talp-upc.gitbook.io/freeling-4-1-user-manual/tagsets. [Último acceso: 02 2019].

[6] «Penn Treebank P.O.S. Tags,» [En línea]. Available: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html. [Último acceso: 02 2019].

[7] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard y D. McClosky, «The Stanford Core Natural Language Processing Toolkit,» *Association of Computer Linguistics,* pp. 55-60, 2014.

[8] S. Brody y N. Diakopoulos, «Coooooooooooooollllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs,» *Empur. Methods Nat. Lang. Process.,* pp. 562-570, 2011.

[9] J. Vivaldi, «Catálogo de herramientas informátivas relacionadas con la creación, gestión y explotación de corpus textuales,» *Revista Tradumàtica - Traducció i Tecnologies de la Informació i la Comunicació,* nº 07, p. http://www.fti.uab.cat/tradumatica/revista, 2009.

[10] B. Valverde Olmedo y R. Martinez García, «Herramientas de código abierto para el procesamiento del español. Análisis y comparativa,» *Dialnet. Revista hiberoamericana de lingüística,* nº 11, pp. 287-308, 2016.

[11] E. D. Liddy, Natural Language Processing, 2001.

[12] B. Felbo, A. Mislove, A. Sogaard, I. Rahwan y S. Lehmann, Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm, 2017.

[13] M. P. Marcus, M. A. Marcinkiewicz y B. Santorini, «Building a Large Annotated Corpus of English: The Penn Treebank,» *Journal Computational Linguistics,* vol. 19, pp. 313-330, 1993.

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 3 – Etiquetado Morfosintáctico *(POS Tagging)*

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Problema del día

¿Cómo identificar la función de una palabra en una oración con técnicas estadísticas?

Encuesta previa

- ¿Sabes enunciar la definición de probabilidad de transición?
- ¿Sabes enunciar la definición de probabilidad de emisión?
- ¿En cuál de las dos influye el corpus lingüístico con el que se trabaja?

En el día de hoy

- Repaso de la sesión anterior
- Algoritmo de Viterbi
- Etiquetado basado en Machine Learning
- Named-Entity Recognition

Repaso

- Nuestro objetivo: POS tagging

- Nuestro modelo: Hidden Markov Models

- Nuestros ingredientes:
  - Un conjunto de estados posibles (etiquetas)
  - Una observación (oración a etiquetar)
  - Matriz de probabilidades de transición
  - Matriz de probabilidades de emisión

Repaso

- Cosas que hemos asumido como ciertas:
  - La probabilidad de aparición de una **palabra** depende solo de **su propia etiqueta** y es independiente de las palabras y etiquetas vecinas.
  - La probabilidad de una **etiqueta** solo depende de la **etiqueta anterior**, en lugar de toda la secuencia de etiquetas

Repaso

- Cosas que hemos asumido como ciertas:
  - La probabilidad de aparición de una **palabra** depende solo de **su propia etiqueta** y es independiente de las palabras y etiquetas vecinas.

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i^n|t_i^n)$$

  - La probabilidad de una **etiqueta** solo depende de la **etiqueta anterior**, en lugar de toda la secuencia de etiquetas

$$Pt_i^n \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Algoritmo de Viterbi: objetivo

- ¿Cuáles son las categorías gramaticales de cada palabra en la siguiente oración?

El señor vino tarde

Otra formulación, más ajustada a la computación:

- ¿Cuál es la secuencia de etiquetas más probable para la secuencia de palabras observada?

Algoritmo de Viterbi: observación

- El
- Señor
- Vino
- Tarde

Algoritmo de Viterbi: conjunto de etiquetas

- El
  - Artículo Determinado
- Señor
  - Sustantivo común
  - Sustantivo propio
  - Adjetivo
- Vino
  - Sustantivo común
  - Verbo
- Tarde
  - Sustantivo común
  - Adverbio
  - Verbo

– Artículo Determinado	– DT
– Sustantivo común	– NN
– Sustantivo propio	– NNP
– Adjetivo	– JJ
– Verbo	– VBD
– Adverbio	– RB

https://www.buscapalabra.com/categoria-gramatical-tiempo-verbal.html

Algoritmo de Viterbi: matriz de probabilidad de transición

Valores del WSJ Corpus

	DT	NN	NNP	JJ	VBD	RB
DT	0,0017	0,4744	0,1147	0,2157	0,0002	0,0102
NN	0,0068	0,1216	0,0096	0,0086	0,0314	0,0177
NNP	0,0025	0,0584	0,3777	0,0084	0,0009	0,0090
JJ	0,0036	0,4509	0,0366	0,733	0,0001	0,0036
VBD	0,2231	0,0615	0,0322	0,0837	0,0050	0,0514
RB	0,0479	0,0120	0,0068	0,1012	0,1011	0,0728
<start>	0,2026	0,0449	0,2767	0,0453	0,0031	0,0510

Algoritmo de Viterbi: matriz de probabilidad de emisión

(valores ficticios, en un caso real son mucho más pequeños)

	El	Señor	Vino	Tarde
DT	0,5	0	0	0
NN	0	0,2	0,1	0,05
NNP	0	0,15	0	0
JJ	0	0,1	0	0
VBD	0	0	0,4	0,2
RB	0	0	0	0,4

Algoritmo de Viterbi: valores de viterbi

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij} b_j(o_t)$$

	<start>	El	Señor	Vino	Tarde	<end>
<end>						
DT						
NN						
NNP						
JJ						
VBD						
RB						
<start>						

- *Máximo de: valor de viterbi previo * p.transición * p.emisión*

Algoritmo de Viterbi: valores de viterbi

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0	0	
DT		0,1013	0	0	0	
NN		0	0,00961134	0,00011687	7,1059E-07	
NNP		0	0,00174287	0	0	
JJ		0	0,00218504	0	0	
VBD		0	0	0,00012072	7,3397E-07	
RB		0	0	0	2,482E-06	
<start>	1	0	0	0	0	

Algoritmo de Viterbi: ruta de máxima probabilidad

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0	0	
DT		0,1013	0	0	0	
NN		0	0,00961134	0,00011687	7,1059E-07	
NNP		0	0,00174287	0	0	
JJ		0	0,00218504	0	0	
VBD		0	0	0,00012072	7,3397E-07	
RB		0	0	0	2,482E-06	
<start>	1	0	0	0	0	

Algoritmo de Viterbi: ruta de máxima probabilidad

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0	0	
DT		0,1013	0	0	0	
NN		0	0,00961134	0,00011687	7,1059E-07	
NNP		0	0,00174287	0	0	
JJ		0	0,00218504	0	0	
VBD		0	0	0,00012072	7,3397E-07	
RB		0	0	0	2,482E-06	
<start>	1	0	0	0	0	

Algoritmo de Viterbi: ruta de máxima probabilidad

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0	0	
DT		0,1013	0	0	0	
NN		0	0,00961134	0,00011687	7,1059E-07	
NNP		0	0,00174287	0	0	
JJ		0	0,00218504	0	0	
VBD		0	0	0,00012072	7,3397E-07	
RB		0	0	0	2,482E-06	
<start>	1	0	0	0	0	

Algoritmo de Viterbi: ruta de máxima probabilidad

	<start>	El	Señor	Vino	Tarde	<end>
<end>		0	0	0	0	
DT		0,1013	0	0	0	
NN		0	0,00961134	0,00011687	7,1059E-07	
NNP		0	0,00174287	0	0	
JJ		0	0,00218504	0	0	
VBD		0	0	0,00012072	7,3397E-07	
RB		0	0	0	2,482E-06	
<start>	1	0	0	0	0	

El/DT señor/NN vino/VBD tarde/RB

Etiquetado basado en Machine Learning

- Diferencia con respecto a otros ámbitos (señal, imagen, datos, etc)

¿Cómo representamos el texto a la entrada?

Palabras → Vectores

El señor come pescado

Como categorías (one-hot)

El	señor	Come	Pescado
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Bag of Word

- Recuento
- Frecuencia relativa
- TF-IDF

Word embeddings

- Word2Vec
- Glove

Métodos pre-entrenados que producen vectores de menor dimensionalidad (aprox. 300)

Etiquetado basado en Machine Learning

- Diferencia con respecto a otros ámbitos (señal, imagen, datos, etc)

¿Cómo representamos la secuencialidad?

Etiquetado basado en Machine Learning

Chiche, A., Yitagesu, B. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *J Big Data* **9**, 10 (2022). https://doi.org/10.1186/s40537-022-00561-y

Named Entity Recognition

- Definición

Reconocimiento de entidades con nombre (NER) es una de las características que ofrece Azure Cognitive Service for Language, una colección de algoritmos de aprendizaje automático e inteligencia artificial en la nube para el desarrollo de aplicaciones inteligentes en las que interviene lenguaje escrito. La característica Reconocimiento de entidades con nombre puede identificar y clasificar entidades en texto no estructurado. Por ejemplo: personas, lugares, organizaciones y cantidades. (Microsoft)

Busca localizar y clasificar en categorías ***predefinidas****, como personas, organizaciones, lugares, expresiones de tiempo y cantidades, las entidades nombradas encontradas en un texto. (Wikipedia)*

Named Entity Recognition

- Categorías predefinidas
  - **PER**: personas, como por ejemplo el nombre de alguien (ej.: Frodo Baggins)
  - **GPE**: ubicaciones geopolíticas: países, ciudades o estados (ej.: Madrid).
  - **LOC**: ubicaciones concretas no geo-políticas (ej.: Vesubio).
  - **ORG**: organizaciones o empresas (ej.: Microsoft).
  - **MONEY**: referencias a dinero (ej.: $ 5).
  - **DATE:** fechas (ej.: 05/02/2021 o «jueves»).
- Categorías específicas
  - P.ej: book-name; weapon; hardware;

Named Entity Recognition

- Ejemplo
  - “En un lugar de La Mancha, de cuyo nombre no quiero acordarme…”
  - “Yo he visto cosas que vosotros no creeríais. Atacar naves en llamas más allá de Orión. He visto rayos-C brillar en la oscuridad cerca de la Puerta de Tannhäuser. Todos esos momentos se perderán en el tiempo, como lágrimas en la lluvia. Es hora de morir.”

Named Entity Recognition

- Técnicas
  - Goyal, A., Gupta, V., & Kumar, M. (2018). Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, *29*, 21-43.
  - Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 50-70.
- Sistemas basados en reglas
- Aprendizaje automático

En resumen

- Algoritmo de Viterbi
  - Cálculo de los valores de Viterbi, y luego vuelta atrás
  - Depende del corpus que ha dado lugar a las prob. de transmisión y emisión
  - Se hacen suposiciones para simplificar

- POS Basado en machine learning
  - Reto: representar adecuadamente el texto a la entrada de la red
  - Apendizaje supervisado

- Named-Entity Recognition
  - Identificar fragmentos de texto como categorías pre-establecidas

En la próxima sesión

¿Cómo construir el árbol sintáctico de una oración?

Y además: Presentación de la Actividad 1.

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 4 – Análisis Sintáctico

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Problema del día

¿Cómo construir el árbol sintáctico de una oración?

(y Presentación de la Actividad 1)

Encuesta previa

- ¿Qué tal vais con la entrega de la actividad individual?

- ¿Sabes qué es el árbol sintáctico de una oración?

En el día de hoy

- Sintaxis, análisis sintáctico
- Gramáticas de estructura sintagmática
- Métodos para el análisis sintáctico basado en programación dinámica

De donde venimos…

El señor vino tarde

- Análisis morfológico
  - el Definite=Def|Gender=Masc|Number=Sing|PronType=Art
  - señor Gender=Masc|Number=Sing
  - venir Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin
  - tarde

- Análisis morfosintáctico (POS)
  - DET
  - NOUN
  - VERB
  - ADV

A dónde vamos

https://nlp.lsi.upc.edu/freeling/demo/demo.php

O
SN
SV
DET
N
V
CC
El señor vino tarde

El análisis sintáctico

- En el análisis sintáctico se determinan las relaciones estructurales entre palabras (apuntes de la asignatura)

- El análisis sintáctico es, en el campo de la lingüística, el análisis de las **funciones sintácticas** o relaciones de **concordancia** y **jerarquía** que guardan las palabras agrupándose entre sí en **sintagmas** u **oraciones**. (www.wikilengua.org)

El análisis sintáctico

- En el análisis sintáctico se determinan las relaciones estructurales entre palabras (apuntes de la asignatura)

- El análisis sintáctico es, en el campo de la lingüística, el análisis de las **funciones sintácticas** o relaciones de **concordancia** y **jerarquía** que guardan las palabras agrupándose entre sí en **sintagmas** u **oraciones**. (www.wikilengua.org)

El árbol sintáctico

- Oración: conjunto de sintagmas
- Sintagma (o grupo): constituyente sintáctico formado por dos o más constituyentes sintácticos
- Nodos: constituyentes sintácticos
- Hojas: palabras de la oración

Gramáticas de estructura sintagmática

- Formalizan, con el objetivo de llevar a cabo tratamiento computacional, el conocimiento de la gramática.

  - Vocabulario terminal **T**
  - Vocabulario no terminal **N**
    - **V** es la unión entre **T** y **N**

  - Conjunto de reglas **R**
  - Símbolo inicial **I**

Gramáticas de estructura sintagmática

- Ejemplo:

- **T** = {j,a}
- **N** = {J, A, E}
- **R** = {
  - E→ JA
  - J → j
  - A → a
  - A → Aa
  - E → EE

  }
- **I** = E

- jajajaa
  - **jajajaa**
  - Jajajaa
  - JAjajaa
  - Ejajaa
  - EJajaa
  - EJAjaa
  - EEjaa
  - EEJaa
  - EEJAa
  - EEJA
  - EEE
  - EE
  - E

Relaciones estructurales

Son reglas para la formación de sintagmas, o de oraciones

- Ejemplos:

SN → Det N

Se lee: El **sintagma nominal** se puede formar con un **determinante** y un **nombre**, en ese orden.

O → SN SV

Se lee: Una oración se puede formar con un **sintagma nominal** seguido de un **sintagma verbal**, en ese orden.

Gramáticas de estructura sintagmática

T = {la, una, cultura, literatura, novedad, árabe, griega, moderna}
N = {D,N,A,GA,GN,SN}
R = {D → {la, una}
N → {cultura, literatura, novedad}
A → {árabe, griega, moderna}
GA → A
GA → A GA
GN → N GA
SN → D GN}
I = SN

- una cultura → D N
- la literatura moderna → D N A → D N GA → D GN → SN

Tipos de gramáticas

- De estados finitos
  - En el lado izquierdo, símbolo NO TERMINAL
  - En el lado derecho ≥ 1 , por lo menos uno debe ser terminal
- **Libres de contexto**
  - En el lado izquierdo, símbolo NO TERMINAL
  - En el lado derecho ≥ 0 , terminales o no terminales
- Sensibles al contexto
  - Número de símbolos izda. $\leq$ Número de símbolos dcha.
- Enumerables recursivamente
  - Sin restricciones

Tipos de gramáticas

- Con categorías complejas

T ={el, la, los, las, un, uno, una, unas, unos, señor, señora, gato, gata, ciudad, vino, viajó, comió, tarde, pronto}

N = {SN, SV, DET, N, Vt, Vi, CC}

R =

- DET → {el, la, los, las, un, uno, una, unas, unos}
- N → {señor, señora, gato, gata, ciudad}
- Vt → {comió}
- Vi → {viajó, vino}
- CC → {tarde, pronto}
- O → SN SV
- SN → N
- SN → DET N
- SV → Vi CC
- SV → Vi
- SV → Vt SN

I = O

La señor vino tarde

El señores vino tarde

El gata vino tarde

Tipos de gramáticas

- Con categorías complejas

T ={el, la, los, las, un, uno, una, unas, unos, señor, señora, gato, gata, ciudad, vino, viajó, comió, tarde, pronto}

N = {SN, SV, DETms, DETmp, DETfs, DETfp, Nms, Nfs, Vt, Vi, CC}

I = O

R =

- **DETms → {el, un, uno}**
- **DETfs → {la, una}**
- **DETmp → {los, unos}**
- **DETfp → {las,unas}**
- **Nms → {señor, gato}**
- **Nfs → {señora, gata, ciudad}**
- Vt → {comió}
- Vi → {viajó, vino}
- CC → {tarde, pronto}
- O → SN SV
- SN → N
- **SN → DETms Nms**
- **SN → DETfs Nfs**
- SV → Vi CC
- SV → Vi
- SV → Vt SN

Tipos de gramáticas

- Con categorías complejas

T ={el, la, los, las, un, uno, una, unas, unos, señor, señora, gato, gata, ciudad, vino, viajó, comió, tarde, pronto}

N = {SN, SV, DETms, DETmp, DETfs, DETfp, Nms, Nfs, Vt, Vi, CC}

I = O

R =

- **DETms → {el, un, uno}**
- **DETfs → {la, una}**
- **DETmp → {los, unos}**
- **DETfp → {las,unas}**
- **Nms → {señor, gato}**
- **Nfs → {señora, gata, ciudad}**
- Vt → {comió}
- Vi → {viajó, vino}
- CC → {tarde, pronto}
- O → SN SV
- SN → N
- **SN → DETms Nms**
- **SN → DETfs Nfs**
- SV → Vi CC
- SV → Vi
- SV → Vt SN

Tipos de gramáticas

- Con categorías complejas

El = {DET, m, s}

La = {DET, f, s}

…

Señor = {N, m, s}

…

N = {SN, SV, DET, N, Vt, Vi, CC}

I = O

R =

- O → SN [num= NUM, gen=GEN] SV [num= NUM, gen=GEN]
- SN [num= NUM, gen=GEN] → N [num= NUM, gen=GEN]
- **SN** [num= NUM, gen=GEN] →

  DET [num= NUM, gen=GEN] N [num= NUM, gen=GEN]
- …

Tipos de gramáticas

- **Libres de contexto**
  - En el lado izquierdo, símbolo NO TERMINAL
  - En el lado derecho ≥ 0 , terminales o no terminales

- **Forma normal de Chomsky**
  - Una gramática formal está en **Forma normal de Chomsky** si todas sus reglas de producción son de alguna de las siguientes formas
    - A → a
    - A → BC
  - Cualquier gramática libre de contexto se puede transformar en su CNF

Algoritmo Cocke-Kasami-Younger (CKY)

- Conceptos básicos
  - Determina si una secuencia pertenece al lenguaje definido en una GLC
  - Requiere que la GLC esté en forma normal de Chomsky
    - Toda GLC se puede escribir en forma normal
  - Destaca por su eficiencia

Algoritmo Cocke-Kasami-Younger (CKY)

- Pasos

1. Transformar la gramática a su forma normal de Chomsky
2. Construir una matriz triangular a partir de la oración
3. Rellenar la matriz
4. Reconstruir el árbol sintáctico

Algoritmo Cocke-Kasami-Younger (CKY)

1. Transformar la gramática a su forma normal de Chomsky

Algoritmo Cocke-Kasami-Younger (CKY)

1. Transformar la gramática a su forma normal de Chomsky

Lexicón:

N = {gato, pescado, perro, pez, ..}

Det = {el, la,una,…}

Vi = {salta, duerme, …}

Vt = { ve, ama, come,…}

Reglas:

1) O -> SN SV

2) SN -> N

3) SN -> Det N

4) SV -> Vi

5) SV -> Vt SN

Gramática en CNF

1) O -> SN SV

2) SN -> gato | pescado | perro | pez

3) Det -> el | la | una

4) SN -> Det N

5) N -> gato | pescado | perro | pez

9) SV -> salta | duerme

10) Vt -> ve | ama | come

11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

2. Construir una matriz triangular a partir de la oración

El	gato	come	pescado
(0,1)	(0,2)	(0,3)	(0,4)
	(1,2)	(1,3)	(1,4)
		(2,3)	(2,4)
			(3,4)

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1)	(0,2)	(0,3)	(0,4)
	(1,2)	(1,3)	(1,4)
		(2,3)	(2,4)
			(3,4)

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2)	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4)
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4)
		(2,3) Vt	(2,4) SV
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4)
	(1,2) SN N	(1,3)	(1,4) O
		(2,3) Vt	(2,4) SV
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

3. Rellenar la matriz

El	gato	come	pescado
(0,1) Det	(0,2) SN	(0,3)	(0,4) O
	(1,2) SN N	(1,3)	(1,4) O
		(2,3) Vt	(2,4) SV
			(3,4) SN N

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

4. Reconstruir el árbol

Gramática en CNF

1) O -> SN SV
2) SN -> gato | pescado | perro | pez
3) Det -> el | la | una
4) SN -> Det N
5) N -> gato | pescado | perro | pez
9) SV -> salta | duerme
10) Vt -> ve | ama | come
11) SV -> Vt SN

Algoritmo Cocke-Kasami-Younger (CKY)

Ambigüedad

"Book the flight through Houston"

Algoritmo CKY probabilístico

- Gramática con probabilidades
- El cálculo de las celdas arrastra la probabilidad

El	gato	come	pescado
(0,1) Det 0.55	(0,2) SN 0.077	(0,3)	(0,4) O 0.00111
	(1,2) N 0.2 SN 0.2	(1,3)	(1,4) O 0.00288
		(2,3) Vt 0.2	(2,4) SV 0.013
			(3,4) SN 0.15 N 0.15

0.7*0.55*0.2

Gramática en CNF

O -> SN SV (0.8)
SN -> gato (0.2)
SN -> pescado (0.15)
Det -> el (0.55)
SN -> Det N (0.7)
N -> gato (0.2)
N -> pescado (0.15)
SV -> salta (0.35)
SV -> duerme (0.27)
Vt -> ve (0.7)
Vt -> ama (0.1)
Vt -> come (0.2)
SV -> Vt SN (0.6)

En la próxima sesión

¿Cómo representar la semántica de las palabras y realizar razonamientos?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 7 – Análisis Semántico

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo representar la semántica de las palabras y realizar razonamientos?

Encuesta previa

- ¿Conoces DBPedia?

- (P1) Si llueve, la calle se moja.

  (P2) No llovió.

  (C) La calle no está mojada.

  - ¿Te suena?

- ¿Conoces la Web Semántica?

En el día de hoy

- Utilidad del análisis semántico
- Representación formal del significado
  - Representación formal con lógicas descriptivas
  - Representación formal con lógicas de primer orden
- Gramáticas Libres de Contexto con anotaciones semánticas
  - Análisis semántico dirigido por la sintaxis

Utilidad del análisis semántico

- Convertir el conocimiento contenido en un texto en un modelo formal con el que poder realizar operaciones lógicas

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ex felis, fermentum eu cursus a, semper nec est. Morbi porttitor.

Representación del conocimiento

- Debe ser verificable, inequívoca, expresiva y permitir la inferencia de nuevo conocimiento.

- Algunas representaciones
  - Lógica de primer orden
    - Lógicas descriptivas
  - Redes Semánticas
  - Frames

Lógicas descriptivas

- Subconjunto de la expresividad de la lógica de primer orden

- Mecanismos para:
  - generar descripciones de conceptos
  - introducir axiomas
  - introducir propiedades de los individuos
  - Inferir nuevos conocimientos a partir de los axiomas

Formalismo de las lógicas descriptivas

- No hay un formalismo único, el término lógica descriptiva denota a una familia de formalismos.
- El más utilizado es ALC.
- De las lógicas descriptivas derivan OWL y la Web Semántica

Formalismo de tripletas

- Ibañez es Autor
- Ibañez escribió El sulfato atómico
- Los autores son personas

Cormenzana, Roberto & López-Borrull, Alexandre. (2018). ESTUDIO DE LA ADAPTACIÓN A RDA Y BIBFRAME EN EL ÁMBITO DE LAS BIBLIOTECAS ESPAÑOLAS: ESTUDIOS DE CASO Title: STUDY OF THE ADAPTATION TO RDA AND BIBFRAME IN THE FIELD OF SPANISH LIBRARIES: CASE STUDIES. Anales de Documentacion. 21. 10.6018/analesdoc.21.2.323441.

Operaciones con lógicas descriptivas

- Subsunción
  - Determinar si una categoría es un subconjunto de otra
- Clasificación
  - Determinar si un individuo pertenece a una categoría

Operaciones con lógicas descriptivas

- Subsunción
  - Determinar si una categoría es un subconjunto de otra

- Clasificación
  - Determinar si un individuo pertenece a una categoría

Lógica de primer orden

- Es un sistema formal diseñado para estudiar la inferencia en los lenguajes de primer orden. (*Wikipedia*)

- Recogemos el **significado** de una oración en lenguaje natural y lo representamos de manera formal, para poder hacer operaciones matemático/lógicas.

- El formalismo es fijo, las variables y funciones son **arbitrarias**.

Ejemplos de la lógica de primer orden

- Luis es profesor
  - Profesor(Luis)

- Luis es profesor de Jorge
  - ProfesorDe(Luis, Jorge)
  - ProfesorDe(Luis, Jorge) ^ ProfesorDe(Luis, Juan)

- Los alumnos son personas
  - ∀ (x) Alumno(x) → Persona(x)

- Existe un profesor que es una persona
  - ∃ (x) Profesor(x) → Persona(x)

Formalismo de la lógica de primer orden

$$
\begin{array}{rcl}
\textit{Sentencia} & \rightarrow & \textit{SentenciaAtómica} \\
 & | & (\textit{Sentencia Conectiva Sentencia}) \\
 & | & \textit{Cuantificador Variable}\ldots\ \textit{Sentencia} \\
 & | & \neg\textit{Sentencia} \\
 & & \\
\textit{SentenciaAtómica} & \rightarrow & \textit{Predicado}(\textit{Término}\ldots) \mid \textit{Término} = \textit{Término} \\
 & & \\
\textit{Término} & \rightarrow & \textit{Función}(\textit{Término}\ldots) \\
 & | & \textit{Constante} \\
 & | & \textit{Variable} \\
 & & \\
\textit{Conectiva} & \rightarrow & \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
\textit{Cuantificador} & \rightarrow & \forall \mid \exists \\
\textit{Constante} & \rightarrow & A \mid X_1 \mid \textit{Juan} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicado} & \rightarrow & \textit{AntesDe} \mid \textit{TieneColor} \mid \textit{EstáLLoviendo} \mid \cdots \\
\textit{Función} & \rightarrow & \textit{Madre} \mid \textit{PiernaIzquierda} \mid \cdots
\end{array}
$$

Bases para el análisis semántico

- Una forma para representar el conocimiento
  - Lógicas descriptivas, lógica de primer orden

- Una forma para, a partir del lenguaje natural, saber qué conocimiento representar
  - Técnicas de extracción del conocimiento → análisis semántico

Bases para la extracción del conocimiento

- Principio de composición (y cómo aplicarlo)

- Hipótesis rule-to-rule (y cómo aplicarla)

Principio de composición

El significado de una oración se construye a partir del significado de sus partes

$$¿\ sgdo(oracion) = \sum sgdo(palabras)?$$

Principio de composición

El significado de una oración se construye a partir del significado de sus partes

$$¿\ sgdo(oracion) = \sum sgdo(palabras)?$$

Ejemplos:

- Análisis de sentimiento con búsqueda de palabras relevantes
- Métodos basados en BoW

Principio de composición

El significado de una oración se construye a partir del significado de sus partes

$$\text{¿}\ \cancel{sgdo(oracion) = \sum sgdo(palabras)}\ ?$$

María ama a Pedro ≠ Pedro ama a María ≠ Pedro María ama a

$$sgdo(oracion) = \sum \{orden, relaciones, agrupacion\}$$

Principio de composición

$$sgdo(oracion) = \sum \{orden, relaciones, agrupacion\}$$

El significado de una oración se construye a partir del significado de sus partes

$$\exists e\ Amar(e) \wedge Amante(e, Pedro) \wedge Amada(e, Maria)$$

Principio de composición

$$sgdo(oracion) = \sum \{orden, relaciones, agrupacion\}$$

El significado de una oración se construye a partir del significado de sus partes

Pedro ama a María → $\exists e\ Amar(e) \wedge Amante(e, Pedro) \wedge Amada(e, Maria)$

María ama a Pedro → $\exists e\ Amar(e) \wedge Amante(e, María) \wedge Amado(e, Pedro)$

Pedro María ama a → $Sin\ significado\ válido$

Hipótesis rule-to-rule

- Cada regla aplicada para constituir el análisis sintáctico tiene un significado equivalente en el análisis semántico

Gramáticas Libres de Contexto anotadas

a) NP → Matías $\{\lambda m.m(Matías)\}$

b) N → restaurante $\{\lambda r.Restaurante(r)\}$

c) Det → un $\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$

d) V_t → abrió $\{\lambda w.\lambda z.w(\lambda x.\exists e\ Abierto(e) \wedge$

$PersonaQueAbre(e,z) \wedge CosaAbierta(e,x)\}$

1) O → SN SV $\{SN.sem\ (SV.sem)\}$

2) SN → NP $\{NP.sem\}$

3) SN → Det N $\{Det.sem\ (N.sem)\}$

4) SV → V_t SN $\{Vt.sem\ (SN.sem)\}$

Se construyen con la hipótesis rule-to-rule

Rule-to-rule en anotaciones semánticas

- Asociar expresiones del cálculo lambda complejas, similares a una función, a las reglas léxicas.
- Copiar el valor semántico del único constituyente a la construcción cuando la regla gramatical solo tiene un constituyente.
- Aplicar la semántica de uno de los constituyentes de una regla gramatical a la semántica de otro de los constituyentes de la regla como si fuera una función.

- Foreach regla en GLC:
  - If (regla tiene símbolo termina a la derecha)
    - Lo rellenas directamente
  - Else if (regla tiene un único símbolo no terminal a la derecha)
    - Copias el símbolo terminal que va debajo
  - Else if (regla tiene dos o más símbolos no terminales a la derecha)
    - Haces una relación de subordinación

Rule-to-rule en anotaciones semánticas

Asociar expresiones del cálculo lambda complejas, similares a una función, a las reglas léxicas.

- NP → Matías $\{ \lambda m.m\ (Matías) \}$
- N → restaurante $\{ \lambda r.Restaurante(r) \}$
- Det → un $\{ \lambda P.\ \lambda Q.\exists x\ P(x) \wedge Q(x) \}$
- V_t → abrió $\{ \lambda w.\ \lambda z.w\ (\lambda x.\ \exists e\ Abierto(e) \wedge PersonaQueAbre(e,z) \wedge CosaAbierta(e,x)) \}$
- O → SN SV
- SN → NP
- SN → Det N
- SV → V_t SN

Rule-to-rule en anotaciones semánticas

Copiar el valor semántico del único constituyente a la construcción cuando la regla gramatical solo tiene un constituyente.

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV
- SN → NP { *NP.sem* }
- SN → Det N
- SV → V_t SN

Rule-to-rule en anotaciones semánticas

- Aplicar la semántica de uno de los constituyentes de una regla gramatical a la semántica de otro de los constituyentes de la regla como si fuera una función

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { *SN.sem(SV.sem)* }
- SN → NP { *NP.sem* }
- SN → Det N { *Det.sem(N.sem)* }
- SV → V_t SN { *V_t.sem(SN.sem)* }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Matías abrió un restaurante

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { *SN.sem(SV.sem)* }
- SN → NP { *NP.sem* }
- SN → Det N { *Det.sem(N.sem)* }
- SV → V_t SN { *V_t.sem(SN.sem)* }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías *{ λm.m (Matías) }*
- N → restaurante *{ λr.Restaurante(r) }*
- Det → un *{ λP. λQ.∃x P(x) ^ Q(x) }*
- Vt → abrió *{ λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV *{ SN.sem(SV.sem) }*
- SN → NP *{ NP.sem }*
- SN → Det N *{ Det.sem(N.sem) }*
- SV → V_t SN *{ V_t.sem(SN.sem) }*

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { *λm.m (Matías)* }
- N → restaurante { *λr.Restaurante(r)* }
- Det → un { *λP. λQ.∃x P(x) ^ Q(x)* }
- Vt → abrió { *λw. λz.w (λx. ∃e Abierto(e)*
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { *SN.sem(SV.sem)* }
- SN → NP { *NP.sem* }
- SN → Det N { *Det.sem(N.sem)* }
- SV → V_t SN { *V_t.sem(SN.sem)* }

{ *Det.sem (N.sem)* }

{ *λP. λQ.∃x P(x) ^ Q(x) (λr.Restaurante(r))* }

{ *λQ.∃x λr.Restaurante(r)(x) ^ Q(x)* }

{ λQ.∃x Restaurante(x) ^ Q(x) }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { λm.m (Matías) }
- N → restaurante { λr.Restaurante(r) }
- Det → un { λP. λQ.∃x P(x) ^ Q(x) }
- Vt → abrió { λw. λz.w (λx. ∃e Abierto(e)
  ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { SN.sem(SV.sem) }
- SN → NP { NP.sem }
- SN → Det N { Det.sem(N.sem) }
- SV → V_t SN { V_t.sem(SN.sem) }

{ V_t.sem(SN.sem) }

{λw. λz.w (λx. ∃e Abierto(e)
^PersonaQueAbre(e,z)^CosaAbierta(e,x)) (λQ.∃x Restaurante((x) ^ Q(x))}

{λz. ∃x Restaurante(x) ^ λx. ∃e Abierto(e)
^PersonaQueAbre(e,z)^CosaAbierta(e,x) (x) }

{ λz. ∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e,z) ^ CosaAbierta(e,x) }

Análisis Semántico en paralelo con el Sintáctico

- NP → Matías { λm.m (Matías) }
- N → restaurante { λr.Restaurante(r) }
- Det → un { λP. λQ.∃x P(x) ^ Q(x) }
- Vt → abrió { λw. λz.w (λx. ∃e Abierto(e) ^PersonaQueAbre(e,z)^CosaAbierta(e,x)) }
- O → SN SV { SN.sem(SV.sem) }
- SN → NP { NP.sem }
- SN → Det N { Det.sem(N.sem) }
- SV → V_t SN { V_t.sem(SN.sem) }

{ SN.sem(SV.sem) }

{ λm.m (Matías) (
λz. ∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e,z) ^ CosaAbierta(e,x)) }

{λz. ∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e,z) ^ CosaAbierta(e,x) (Matías)}

{∃x Restaurante(x) ^ ∃e Abierto(e) ^ PersonaQueAbre(e, Matías) ^ CosaAbierta(e,x)}

Análisis Semántico en paralelo con el Sintáctico

- Ventajas
  - Descarta ramas inválidas semánticamente hablando
  - Dos análisis en un único cálculo
- Inconvenientes
  - Ineficiente computacionalmente
  - Complejo de construir

En resumen

- El análisis semántico busca formalizar el conocimiento
- Los modelos basados en tripletas
  - son más sencillos
  - permiten la aplicación de reglas lógicas.
- Los modelos basados en lógicas de primer orden
  - Son más complejos
  - Permiten paralelizar análisis sintáctico y semántico
  - Permiten una representación completa de la oración

En la próxima semana

¿Cómo aplicar técnicas de desambiguación del significado?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 6 – Semántica Léxica

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo aplicar técnicas de desambiguación del significado?

En el día de hoy

- Semántica Léxica:
  - Significado de las palabras
  - Relaciones formales entre sentidos de las palabras
  - Recursos para la catalogación de significados, WordNet
  - Desambiguación del sentido de las palabras
  - Relaciones informales entre sentidos de las palabras (similitud)

Significado de las palabras

- Semántica Léxica:
  - Campo de la lingüística que estudia el significado de las palabras y las relaciones de sentido que establecen entre ellas
  - Funciones:
    - Caracterizar el significado
      - Todos podemos dar una definición, pero caracterizar no es trivial
    - Estudiar las relaciones entre palabras
    - Estudiar la variación contextual del significado
      - https://www.youtube.com/watch?v=Xyp7xt-ygy0
      - https://www.hermanotemblon.com/laas-1000-acepciones-de-la-palabra-cojones/

Relaciones formales entre palabras

- Homonimia: homografía, homofonía
- Polisemia

- Sinónimos/antónimos
- Hipónimos/hiperónimos
- Merónimos/holónimos
- Metonimia

Relaciones formales entre palabras

- Homonimia: homografía, homofonía

homónimo, ma

Del lat. *homonȳmus*, y este del gr. ὁμώνυμος *homónymos*.

1. adj. Dicho de una persona o de una cosa: Que, con respecto de otra, tiene el mismo nombre. U. t. c. s.

2. adj. *Ling.* Dicho de una palabra: Que se pronuncia como otra, pero tiene diferente origen o significado muy distante; p. ej., *aya* 'niñera' y *haya* 'árbol'. U. t. c. s. m.

homógrafo, fa

De *homo-*[1] y *-grafo*.

1. adj. *Ling.* Dicho de una palabra: Que tiene la misma grafía que otra. Vino *'bebida' y* vino *'forma del verbo* venir*' son términos homógrafos.* U. t. c. s. m.

Requiere desambiguación

homófono, na

Del gr. ὁμόφωνος *homóphōnos*, de ὁμός *homós* 'igual' y φωνή *phōnḗ* 'sonido'.

1. adj. *Ling.* Dicho de una palabra: Que suena igual que otra, pero que tiene distinto significado y puede tener distinta grafía. U. t. c. s. m. *Aunque se distinguen ortográficamente,* tubo *y* tuvo *son homófonos.*

Genera dificultades al reconocer el lenguaje hablado

Relaciones formales entre palabras

- Polisemia
  - Una palabra es polisémica cuando tiene varios significados
    - Gato
    - Banco
    - Copa
  - La polisemia se aplica a los lemas (p.ej. marcha vs marcha)

Requiere desambiguación

Relaciones formales entre palabras

- Sinónimos

sinónimo, ma

Del lat. *synonȳmus*, y este del gr. συνώνυμος *synónymos*, de συν- *syn-* 'con-' y ὄνομα *ónoma* 'nombre'.

1. adj. *Ling.* Dicho de una palabra o de una expresión: Que, respecto de otra, tiene el mismo significado o muy parecido, como *empezar* y *comenzar*. U. t. c. s. m.

- Antónimos

antónimo, ma

De *anti-* y *-ónimo*.

1. adj. *Ling.* Dicho de una palabra: Que, respecto de otra, expresa una idea opuesta o contraria, como *virtud* y *vicio*, *claro* y *oscuro* o *antes* y *después*. U. t. c. s. m.

Son relaciones binarias

Relaciones formales entre palabras

- Hipónimos/hiperónimos

- A es más general que B
  - Animal es más general que perro
  - Perro es más general que dálmata

- Se dice que:
  - B es hipónimo de A
    - Perro es hipónimo de animal
  - A es hiperónimo de B
    - Perro es hiperónimo de dálmata

Relaciones formales entre palabras

- Merónimos/holónimos

- A es parte de B
  - La nariz forma parte de la cabeza
  - La fosa nasal forma parte de la nariz

- Se dice que:
  - A es merónimo de B
    - Nariz es merónimo de cabeza
  - B es holónimo A
    - Nariz es holónimo de fosa nasal

Relaciones formales entre palabras

- Metonimia
  - Uso de una palabra en lugar de otra, valiéndose de una relación existente
- Causa por su efecto
- Contenedor por el contenido
- Símbolo por la cosa simbolizada
- Lugar por lo que en él se produce
- Marca comercial por objeto de la marca
- Obra por el nombre de su autor
- La parte por el todo
- El todo por la parte
- (…)

Genera sinonimia no catalogada

Recursos para la catalogación de significados

- Diccionarios
  - Recogen los significados de un lema

- Tesauros
  - Recogen las relaciones entre significados de un lema

- https://wordnet.princeton.edu/
- https://www.nltk.org/howto/wordnet.html

Para qué desambiguar?

- Una vez que hemos desambiguado:
  - Podemos generar un **resumen sintético**
  - Podemos generar una **traducción**
  - Podemos identificar la **respuesta a una pregunta**, bien con **razonamiento lógico** o bien con ***Information Retrieval***
  - Podemos afinar más en la **similitud entre oraciones**

Antes de seguir, el concepto de “colocación”

- Dada una palabra cualquiera, en un texto cualquiera, la colocación de dicha palabra es la secuencia de N palabras previas, y N palabras posteriores, a la palabra que genera la colocación.

- Es decir, un N-grama generado por las palabras anteriores y posteriores

Métodos basados en conocimiento

- Lesk

Leemos el significado en el diccionario

- Lesk simplificado
  - Se lematiza cada uno de los sentidos de una palabra
  - Se lematiza la colocación
  - Se comparan ambas lematizaciones
- Lesk original
  - Se lematiza cada uno de los sentidos de una palabra
  - Se lematiza cada uno de los sentidos de las palabras colocadas
  - Se comparan las lematizaciones

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:
Empresa dedicada a realizar operaciones financieras con el dinero procedente de sus accionistas y de los depósitos de sus clientes.
(empresa, dedicar, realizar, operación,
financiera, dinero, accionista, depósito, cliente)

banco[2]:
Asiento, con respaldo o sin él, en que pueden sentarse dos o más personas.
(asiento, respaldo, poder, sentarse, persona)

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:

(empresa, dedicar, realizar, operación,
financiera, dinero, accionista, depósito, cliente)

banco[2]:

(asiento, respaldo, poder, sentarse, persona)

Contexto:

(prestar, dinero)

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:

(empresa, dedicar, realizar, operación, financiera, dinero, accionista, depósito, cliente)

banco[2]:

(asiento, respaldo, poder, sentarse, persona)

Contexto:

(prestar, dinero)

Lesk simplificado

- Compara la firma de la palabra ambigua (signature), que se corresponde con **su definición en el diccionario**, con las palabras de contexto (context), es decir, con las **palabras vecinas** a la palabra ambigua

*El **banco** le prestó dinero*

banco[1]:

(empresa, dedicar, realizar, operación, financiera, dinero, accionista, depósito, cliente)

1 solapamiento

banco[2]:

(asiento, respaldo, poder, sentarse, persona)

0 solapamientos

Contexto:

(prestar, dinero)

Lesk original

- Compara la firma de la palabra ambigua (signature), que se corresponde con su **definición en el diccionario**, con **las firmas de las palabras** de contexto (context), es decir, con las palabras vecinas a la palabra ambigua

*Me senté en un **banco** del parque*

banco1:
(empresa, dedicar, realizar, operación, financiera, dinero, accionista, depósito, cliente) — 0 solapamiento

banco2:
(asiento, respaldo, poder, sentarse, persona) — 2 solapamientos

Contexto:
sentar: poner a una persona en un asiento apoyado sobre sus nalgas
(poner, persona, asiento, apoyar, nalgas)
parque: Espacio que se dedica a jardines
(espacio,dedicar, jardin)

Métodos basados en conocimiento

- Ventajas
  - Muy explicables
  - Basta con disponer de un recurso básico (diccionario)

- Inconvenientes
  - No trabaja bien con entradas de diccionario muy cortas
    - Ampliar con sinónimos

  - Todas las palabras tienen el mismo peso
    - Aplicar un peso por colocación, POS Tag.

Aprendizaje no supervisado

- Schütze

  Hacemos clústers para cada uno de los sentidos

- Clusterización
  - Calcular el vector de contexto
  - Clusterizar
  - Calcular el centroide

- Desambiguación
  - Menor distancia entre vectores

Schütze

- Clusterización
  - Se calcula un vector de contexto *c* para cada aparición w_i de la palabra *w* en el corpus

    (con un bag of words de la colocación)

  - Se aplica un algoritmo de agrupamiento sobre los vectores de contexto c en un número predefinido de grupos, que definirán cada uno de los sentidos de la palabra w

    (con k-means)

  - Se calcula el centroide de cada grupo. Cada vector de un centroide será el vector representación de un sentido de la palabra w

    (promedio de cada dimensión)

Vectorizar texto: Empezamos con one-hot-encoding

Cada categoría se representa por un vector único

Color
Red
Green
Blue
Green

Red	Green	Blue
1	0	0
0	1	0
0	0	1
0	1	0

Y cada palabra es una categoría

Vectorizar texto: Empezamos con one-hot-encoding

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Hay 27 palabras diferentes → Se representan con vectores de dimensión 27

en	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lugar	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
la	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mancha	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cuyo	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nombre	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
no	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
quiero	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
acordarme	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ha	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mucho	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tiempo	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
que	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
vivía	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
hidalgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
los	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
lanza	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
adarga	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
antigua	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
rocín	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
flaco	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
galgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
corredor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Vectorizar texto: Bag of Words

- Recuento de palabras
  - “antiguo, de un astillero antiguo”

antiguo,	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
BoW TF	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0

Clusterizar vectores

Schütze

- Desambiguación
  - Se calcula un vector de contexto *c* para la aparición de la palabra *w* que queremos desambiguar

    (con un bag of words de la colocación)
  - Se calcula la distancia desde el vector c hasta cada uno de los centroides
  - Se selecciona el centroide más cercano con el sentido correcto

Schütze

- Ventajas
  - No hace falta un corpus etiquetado
  - Conceptualmente sencillo

- Inconvenientes
  - Mucho preprocesamiento para cada palabra
  - Baja explicabilidad, aunque explicable en última instancia

    (podríamos asignar palabras clave, con peso, a cada corpus)

Aprendizaje automático semisupervisado (bootstraping)

- Yarowsky

El corpus que tenemos es pequeño, y nos tenemos que apañar

- Entreno el clasificador con los (pocos) ejemplos que tengo
- Clasifico el resto de las instancias
- Las instancias más sólidamente clasificadas, las incorporo al corpus etiquetado
- Repito hasta que el error sea bajo, o me quede sin ejemplos

En resumen

- La semántica léxica estudia los significados de las palabras
- La semántica léxica trabaja con los lemas
- Los significados se recogen en Diccionarios
- Las relaciones entre palabras se recogen en Tesauros
- Desambiguación como tarea relevante en semántica léxica
  - Métodos basados en conocimiento
  - Métodos basados en vectores

En la próxima semana

¿Cómo calcular la similitud de dos palabras?

Relaciones informales entre palabras

- Métodos de cálculo de similitud
  - Medidas estadísticas
  - Basadas en tesauro
  - Basados en vectores

Similitud mediante tesauro

- Dos palabras son más similares si están cerca en el árbol del tesauro. Es decir, si hay pocos saltos entre ellas

- En la práctica, se utilizan relaciones de hiponimia e hiperonimia

- Entre sentidos: $$sim_{path(c_1,c_2)} = \frac{1}{pathlen(c_1, c_2)}$$

- Entre palabras: $$wordsim(w_1, w_2) = \max_{\substack{c_1 \in senses(w_1) \\ c_2 \in senses(w_2)}} sim(c_1, c_2)$$

Similitud mediante tesauro: ejemplo

- Similitud entre silla y mesa (chair and table)

- S: (n) **table**, tabular array (a set of data arranged in rows and columns) *"see table 1"*
- S: (n) **table** (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs) *"it was a sturdy table"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *has instance*
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*

- S: (n) **chair** (a seat for one person, with a support for the back) *"he put his coat over the back of the chair and sat down"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) seat (furniture that is designed for sitting on) *"there were not enough seats for all the guests"*
      - *direct hyponym* / *full hyponym*
      - *part meronym*
      - *member holonym*
      - *has instance*
      - ***direct hypernym*** / *inherited hypernym* / *sister term*
        - S: (n) furniture, piece of furniture, article of furniture (furnishings that make a room or other area ready for occupancy) *"they had too much furniture for the small apartment"; "there was only one piece of furniture in the room"*

path(silla, mesa) = 3+1 = 4
sim_path(silla, mesa) = 1/4 = 0.25

Similitud mediante tesauro: problemas

- Subjetividad inherente al tesauro
- Valores discretos
- Se asume igual valor para todos los saltos
  - Alternativas:
    - Normalizar las distancias según la profundidad del árbol
    - Asociar distancias diferentes
    - Métodos basados en la teoría de la información (Resnik)

Similitud mediante vectores

- Vectores extraídos como Word Embedings
  - Word2Vec, Glove, Fasttext, ELMO, BETO, BERT

Inciso: ¿qué es un embedding?

- Representación vectorial de una palabra en la que:
  - Los vectores guardan un tipo de relación con la información de la palabra.
  - Los vectores se calculan con técnicas no supervisadas

Male-Female

Verb tense

Country-Capital

Similitud mediante vectores

- Vectores extraídos como Word Embedings
  - Word2Vec, Glove, Fasttext, ELMO, BETO, BERT

- Métricas matemáticas bien conocidas
  - Distancia euclídea
  - Similitud de coseno
  - Pearson
  - Jaccard
  - Levensthein
  - 17 types of similarity and dissimilarity measures used in data science

Similitud mediante vectores

- Ventajas
  - Facilidad de cálculo
  - No hace falta tesauro o corpus etiquetado
  - Muy buenos resultados

- Desventajas
  - Falta de explicabilidad

En resumen (bis)

- La semántica léxica estudia los significados de las palabras
- La semántica léxica trabaja con los lemas
- Los significados se recogen en Diccionarios
- Las relaciones entre palabras se recogen en Tesauros
- Desambiguación como tarea relevante en semántica léxica
  - Métodos basados en conocimiento
  - Métodos basados en vectores
- La similitud es una relación más relajada
  - Métodos basados en tesauro
  - Métodos basados en vectores

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 7 – Modelado estadístico del lenguaje

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo podemos predecir siguiente texto, dado el comienzo de una oración?

Encuesta previa

- ¿Funciona bien el predictor de texto del teclado móvil?
- ¿Funciona bien el reconocimiento del habla del teclado del móvil?
- ¿Sabes qué es el análisis de sentimientos?

En el día de hoy

- Introducción al Modelado del Lenguaje Estadístico
- Modelado de Lenguaje con N-Gramas
  - Cómo crearlos
  - Cómo medirlos
  - Técnicas de suavizado

- Introducción al Modelado del Lenguaje Vectorial
- Modelado de Lenguaje con Vectores
  - BoW
  - TF
  - TF-iDF

Introducción al modelado del lenguaje estadístico

- Qué es
  - Un método para darle forma computacional al texto

- Para qué sirve
  - Teclados predictivos
  - Traducción automática
  - Speech to text
  - Generación de textos

- Cómo se hace?
  - Estadístico

Modelado de Lenguaje con N-Gramas

- Tokens (palabras)
  - [“En”, “un”, “lugar”, “de”, “La_Mancha”]

- **Bigramas**
  - [[“En”, “un”], [“un”,”lugar”],[“lugar”, “de”], [“de”, “La_Mancha”]]

- **Trigramas**
  - [[“En”, “un”, “lugar”], [“un”,”lugar”, “de”], [“lugar”, “de”, “La_Mancha”]]

- **N-gramas**
  - […]

Modelado de Lenguaje con N-Gramas

- ¿Cuál es la probabilidad de que, dada una palabra, aparezca otra palabra concreta?
- **Para un texto de referencia dado**, ¿Cuál es la probabilidad de que, dada una palabra, aparezca otra palabra concreta?

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- ¿Cuál es la probabilidad de que tras la palabra “want” aparezca la palabra to?

$$P\ (to|want) = \frac{Apariciones\ de\ "want\ to"}{Apariciones\ de\ "want"}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- ¿Cuál es la probabilidad de que tras la palabra “want” palabra to?

$$P\ (to|want) = \frac{Apariciones\ de\ "want\ to"}{Apariciones\ de\ "want"}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Las filas y las columnas no suman lo mismo, pero porque es un ejemplo recortado

Modelado de Lenguaje con N-Gramas

- La probabilidad en un bigrama es fácil, ¿y en un trigrama?

$$P\ (to|I\ want) = \frac{Apariciones\ de\ "I\ want\ to"}{Apariciones\ de\ "I\ want"} = \frac{??}{827}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- ¿Y en un N-Grama?

$$P\,(eat|I\ want\ to) = \frac{Apariciones\ de\ "I\ want\ to\ eat"}{Apariciones\ de\ "I\ want\ to"} = \frac{¿?}{¿?}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Modelado de Lenguaje con N-Gramas

- ¿Y en un N-Grama?
- Aplicamos la regla de la cadena
- Y la hipótesis de Markov

$$\boxed{\begin{gathered} P\ (I\ want\ to\ eat) = \\ P(I) \times \\ P(want|I) \times \\ P(to|I\ want) \times \\ P(eat|I\ want\ to) \end{gathered}}$$

Modelado de Lenguaje con N-Gramas

- ¿Y en un N-Grama?
- Aplicamos la regla de la cadena
- Y la hipótesis de Markov

$$\boxed{\begin{gathered} P\ (I\ want\ to\ eat) = \\ P(I) \times \\ P(want|I) \times \\ P(to|want) \times \\ P(eat|to) \end{gathered}}$$

Modelado de Lenguaje con N-Gramas

- Ya somos capaces de:
  - Dada una secuencia, calcular la probabilidad de una siguiente palabra dada
  - Calcular la probabilidad de una secuencia dada

Siempre, en un corpus de referencia

Modelado de Lenguaje con N-Gramas

- Problemas del modelo
  - Hay palabras que no aparecen en el corpus
  - Hay bigramas que no aparecen en el corpus (y todas sus secuencias serán cero)

Técnicas de suavizado

- ¿Por qué el suavizado de modelos?
- Técnica de Laplace y técnica add-k
- Backoff e interpolación

¿Por qué el suavizado de modelos?

- *"El **marrano**, también conocido como el **cochino**, es un animal que ha sido criado por los seres humanos durante siglos por su carne y otros subproductos como la piel y la grasa. El **chancho**, por otro lado, es una palabra que se usa comúnmente en algunos países de América Latina para referirse al **cerdo**. Y por último, el término porcino se refiere a cualquier animal perteneciente a la familia de los cerdos, como los jabalíes y los cerdos salvajes."*

$$¿\ P(cerdos|los) \approx P(cochinos|los)\ ?$$

Técnica de Laplace y técnica add-k

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

+1

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Técnica de Laplace y técnica add-k

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

+k

Backoff e interpolación

- Backoff
  - Se utilizan N gramas cada vez más pequeños, hasta encontrar uno que sea distinto de cero

$$P(eat|I\ want\ to) \approx P(to|I\ want) \approx P(want|I) \approx P(I)$$

- Interpolación
  - Se calcula como combinación lineal de n-gramas de menor orden

$$P(eat|I\ want\ to) \approx \lambda_1 P(to|I\ want) \times \lambda_2 P(want|I) \times \lambda_3 P(I)$$

Evaluación

- Nos dice en qué medida funciona un modelo

- Extrínseca
  - Se prueba en la aplicación final
    (p.ej. *Precission* y *recall* en la predicción de un teclado móvil)

- Intrínseca
  - Perplejidad

Perplejidad

- Cuanto menor sea el valor, mejor es el modelo

$$PP(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}}$$

Introducción al modelado del lenguaje vectorial

- Qué es
  - Un método para darle forma computacional al texto

- Para qué sirve
  - Análisis de sentimientos
  - Topic detection
  - Clasificación de textos

- Cómo se hace?
  - Vectorial

Nuestro texto de referencia

En un lugar de La Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.

Empezamos con one-hot-encoding

Hay 27 palabras diferentes → Se representan con vectores de dimensión 27

en	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lugar	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
la	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mancha	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cuyo	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nombre	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
no	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
quiero	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
acordarme	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ha	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mucho	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tiempo	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
que	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
vivía	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
hidalgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
los	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
lanza	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
adarga	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
antigua	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
rocín	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
flaco	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
galgo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
corredor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Características de one-hot-enconding

- Es fácil de generar y manejar
- Todos los vectores son ortogonales entre sí
- En redes neuronales, cada palabra activaría una neurona de entrada diferente
- Dimensión del vector potencialmente muy elevada
- El vector no contiene información sobre la palabra, es sólo una asignación arbitraria

Modelos de lenguaje con Vectores

- Bag of Words (BoW), es una técnica que genera un único vector que contiene todas las palabras de un texto.

- BoW binario
- BoW TF
- BoW WTF
- BoW TF-iDF

BoW binario

- 1 si la palabra aparece. 0 si la palabra no aparece.
  - “antiguo, de un astillero antiguo”

antiguo,	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
BoW binario	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

BoW TF

- Recuento de palabras
  - “antiguo, de un astillero antiguo”

antiguo,	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
BoW TF	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0

BoW WTF

- Frecuencia de aparición de las palabras
  - “antiguo, de un astillero antiguo”

antiguo,	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
de	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
un	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
astillero	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
antiguo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
BoW WTF	0	0,2	0	0,2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,2	0	0,4	0	0	0	0	0

BoW TF-iDF (representación global)

- Se pondera TF con respecto a las apariciones del mismo término en otros documentos
  - Nos dice si la palabra es un factor diferencial de este texto con respecto a otros

- Ejemplo (no estricto, pero muy ilustrativo)

  https://flowingdata.com/2018/09/18/cuisine-ingredients/

  - En el ejemplo,
    - “most used” sería TF
    - “cuisine-specific” sería TF-iDF

Una vez que tengo el texto como vector

Una vez que tengo el texto como vector

- Problemas
  - Vectores de entrada potencialmente muy grandes
  - Vectores de entrada que NO tienen relación con el significado
  - Modelo incapaz de entender la secuencialidad del texto

En resumen

- Modelos estadísticos
  - Modelo que nos permite conocer la probabilidad de aparición de un N-Grama
  - Aplicaciones
    - Teclados móviles
    - Traducción
    - Text-to-speech

- Modelado con vectores
  - Modelo que nos permite construir un vector que representa a un texto
  - Aplicaciones
    - Procesado neuronal
    - Análisis de sentimientos

En la próxima semana

¿Cómo podemos representar una palabra en forma vectorial?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 8 – Modelado de lenguaje neuronal

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

La pregunta del día

¿Cómo podemos representar una palabra en forma vectorial?

¿Cómo podemos superar las limitaciones de one-hot-encoding?

Encuesta previa

- ¿Qué tal dominas el concepto de red neuronal?
- ¿Qué tal dominas el concepto de aprendizaje supervisado?
- ¿Qué tal dominas el concepto de aprendizaje no supervisado?
- ¿Conoces autoencoders?

En el día de hoy

- Introducción a embedings

- Word2Vec
  - CBOW
  - SkimGram
  - Ejemplo de Word2Vec con código

- Otros Modelos
  - FastText
  - Glove

- Uso de los Embedings

Introducción a embedings

- Problemas de Bag of Words
  - Alta dimensionalidad
    - Una dimensión por cada palabra representada
    - El modelo en_core_web_lg tiene 514K palabras
  - Dispersión
    - En un modelo de 1000 palabras, una frase de 10 palabras tendrá 990 ceros y 10 unos.
  - Falta de relación con el significado de la palabra
    - Todos los vectores ortogonales entre sí, por lo que:
      - su producto escalar es cero
      - Su distancia euclídea es siempre la misma

Introducción a embedings. Idea de partida

- Idea inicial:
  - Reducir dimensionalidad y dispersión
  - Utilizar conceptos derivados de autoencoders

Word2Vec

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

https://arxiv.org/abs/1310.4546

Propuesto por Google en 2013

Word2Vec. CBOW (Continuous Bag of Words)

- Cómo entrenar la red

- "En un lugar de La Mancha, de cuyo nombre no quiero acordarme"
  - "En un lugar de La"
  - "un lugar de La Mancha"
  - "lugar de La Mancha de"
  - "de La Mancha de cuyo"
  - "La Mancha de cuyo nombre"

Word2Vec. Skip-Gram.

- Cómo entrenar la red

- "En un lugar de La Mancha, de cuyo nombre no quiero acordarme"
  - "En un lugar de La"
  - "un lugar de La Mancha"
  - "lugar de La Mancha de"
  - "de La Mancha de cuyo"
  - "La Mancha de cuyo nombre"

Word2Vec. Cómo obtener el embeding

Word2Vec

- Resultados
  - Vectores de dimensión 300
  - No hay dispersión
  - Posición del vector relacionada con el significado
    - https://projector.tensorflow.org/

Posición del vector relacionada con el significado

- Permite establecer similitud entre palabras
  - Campos semánticos de detección automática
  - Mucha ayuda en topic detection

- Permite procesar palabras que previamente no habían aparecido
  - Vectores similares a la entrada darán clasificaciones similares

Representación con vectores de palabras

- Cómo entrenar un modelo con Word2Vec

https://github.com/kavgan/nlp-in-practice/tree/master/word2vec

- Representación tridimensional de Word2Vec

https://projector.tensorflow.org/

Otros Modelos. FastText

- Problema de Word2Vec.
  - Palabras mal escritas
  - Palabras nuevas

- Solución propuesta por FastText
  - Hacer que los tokens sean n-gramas de palabras

    <wh, whe, her, ere, re>

Multiple models, each with different capabilities and price points. Prices are per 1,000 tokens. You can think of tokens as pieces of words, where 1,000 tokens is about 750 words. This paragraph is 35 tokens.

https://openai.com/pricing

Word2Vec vs FastText

```
print(w2v.wv['maravilloso'])
```

```
print(w2v.wv['maravill0so'])
```

(Demo con código)

Otros Modelos. Glove

- Se afronta el problema con herramientas matriciales
  - Reducción de la dimensionalidad con matriz característica

- Word2Vec vs Glove
  - Resultados similares
  - Glove es paralelizable

Uso de Word embedings

- Similitud semántica entre palabras / entre oraciones
- Input para redes neuronales

https://towardsdatascience.com

Uso de Word embedings

- Similitud semántica entre palabras / entre oraciones
- Input para redes neuronales

https://towardsdatascience.com

En la próxima semana

¿Cómo podemos procesar una secuencia de tamaño variable con una red neuronal?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA unir

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 8 – Modelado de lenguaje neuronal

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

En el día de hoy

¿Cómo podemos procesar una secuencia de tamaño variable con una red neuronal?

Encuesta previa

- ¿Cómo funciona una red neuronal?
- ¿Qué es un Word embedding?
- ¿Qué técnica introduce la idea de partir las palabras en tokens menores?

En el día de hoy

- Embedings (recordatorio)

- Redes recurrentes y LSTM
  - ELMO

- Transformers
  - BERT
  - BETO

Embeddings (recordatorio)

- Word2Vec

- Vectores que representan palabras, y guardan relación con su significado
- Generados con entrenamiento no supervisado

Procesamiento con redes clásicas

$$W * w_{onehot.palabra} = w_{emb.palabra}$$

Procesamiento con redes clásicas

$$W * w_{onehot.palabra} = w_{emb.palabra}$$

Procesamiento con redes clásicas

$$W * w_{onehot.palabra} = w_{emb.palabra}$$

Procesamiento con redes clásicas

$$W * w_{onehot.palabra} = w_{emb.palabra}$$

Procesamiento con redes clásicas

$$W * w_{onehot.palabra} = w_{emb.palabra}$$

Redes recurrentes

Fang, Hao & Jiang, Man & Lu, Jingyan & Qin, Gong & Tao, & Tang, Peifu & Yunjun, Yan & Yun, Luo & Liu, Jingtian. (2020). A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages. Remote Sensing. 12. 256. 10.3390/rs12020256.

Redes recurrentes. Sequence to Sequence

Redes recurrentes. Influencia decreciente

Redes LSTM

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

Redes LSTM

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

Redes LSTM bidireccionales

Fu, Jian & Chu, JingChun & Guo, Peng & Chen, Zhenyu. (2019). Condition Monitoring of Wind Turbine Gearbox Bearing Based on Deep Learning Model. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2912621.

ELMO

- Utiliza redes **LSTM bidireccionales** para extraer el embedding
  - El embedding hace mejor uso del contexto

- Una palabra no tiene representación única
  - Un embedding diferente **para una misma palabra** en oraciones diferentes

ELMO demo

- Ejemplo extraído de https://www.geeksforgeeks.org/overview-of-word-embedding-using-embeddings-from-language-models-elmo/

https://colab.research.google.com/drive/1ivu0eVa6uacmjS_gq8P4SDGJR-8bbHhm?usp=sharing

Transformers

Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). **Attention is all you need**. *Advances in neural information processing systems*, *30*.

Vector key, vector query

- Vector key
  - Representa a la palabra
- Vector query
  - Representa las partes de la frase a las que esta palabra presta atención

Yo	[-0.33 -0.44 ... 0.11]
Voy	[-0.76 -0.59 ... -0.40]
A	[0.14 -0.15 ... 0.43]
Aprobar	[-0.08 0.01 ... 0.48]

[0,33	Yo
-0.87	Voy
0,56	A
0,01]	Aprobar

BERT

[CLS]
it
looks
small
outside
,
but
quite
big
inside
nice
styling
too

BERT demo

- Ejemplo extraído de https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

https://colab.research.google.com/drive/1ZQvuAVwA3IjybezQOXnrXMGAnMyZRuPU

BETO: Spanish BERT

- BETO is a BERT model trained on a big Spanish corpus.

Canete, J., Chaperon, G., Fuentes, R., Ho, J. H., Kang, H., & Pérez, J. (2020). **Spanish pre-trained bert model and evaluation data**. *Pml4dc at iclr*, *2020*, 1-10.

Task	BETO-cased	BETO-uncased	Best Multilingual BERT	Other results
POS	**98.97**	98.44	97.10 [2]	98.91 [6], 96.71 [3]
NER-C	**88.43**	82.67	87.38 [2]	87.18 [3]
MLDoc	95.60	**96.12**	95.70 [2]	88.75 [4]
PAWS-X	89.05	89.55	90.70 [8]	
XNLI	**82.01**	80.15	78.50 [2]	80.80 [5], 77.80 [1], 73.15 [4]

BETO: Spanish BERT

- *Spanish Wikis*: Wich include Wikipedia, Wikinews, Wikiquotes and more.
- *ParaCrawl*: Spanish portion of ParaCrawl (http://opus.nlpl.eu/ParaCrawl.php)
- *EUBookshop*: Spanish portion of EUBookshop (http://opus.nlpl.eu/EUbookshop.php)
- *MultiUN*: Spanish portion of MultiUN (http://opus.nlpl.eu/MultiUN.php)
- *OpenSubtitles*: Spanish portion of OpenSubtitles2018 (http://opus.nlpl.eu/OpenSubtitles-v2018.php)
- *DGC*: Spanish portion of DGT (http://opus.nlpl.eu/DGT.php)
- *DOGC*: Spanish portion of DOGC (http://opus.nlpl.eu/DOGC.php)
- *ECB*: Spanish portion of ECB (http://opus.nlpl.eu/ECB.php)
- *EMEA*: Spanish portion of EMEA (http://opus.nlpl.eu/EMEA.php)
- *Europarl*: Spanish portion of Europarl (http://opus.nlpl.eu/Europarl.php)
- *GlobalVoices*: Spanish portion of GlobalVoices (http://opus.nlpl.eu/GlobalVoices.php)
- *JRC*: Spanish portion of JRC (http://opus.nlpl.eu/JRC-Acquis.php)
- *News-Commentary11*: Spanish portion of NCv11 (http://opus.nlpl.eu/News-Commentary-v11.php)
- *TED*: Spanish portion of TED (http://opus.nlpl.eu/TED2013.php)
- *UN*: Spanish portion of UN (http://opus.nlpl.eu/UN.php)

En resumen

- Los embeddings generan **vectores que representan palabras**.
- Las redes secuenciales generan embeddings que tienen en cuenta **el contexto**.
- Una palabra tiene **diferentes representaciones** según el contexto.
- Los Transformers añaden el concepto de **atención**, que amplía aún más el contexto que se puede tener en cuenta.
  - Bert, modelo original, en inglés
  - BETO, modelo pre-entrenado en español

En la próxima semana

¿Cómo podemos procesar un texto de gran tamaño con una red neuronal?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 8 – Modelado de lenguaje neuronal

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

En el día de hoy

¿Cómo podemos procesar un texto de gran tamaño con una red neuronal?

Encuesta previa

- ¿Qué es relevante en una red recurrente?
- ¿Qué diferencia una RNN de una LSTM?
- ¿Cuál es la arquitectura de Sequence2Sequence?
- ¿Qué es un embedding contextual?

En el día de hoy

- embeddings, RNN, LSTM (recordatorio)

- Transformers
  - Atención
  - Posicionamiento

- Modelos preentrenados
  - BERT
  - BETO
  - HuggingFace

Embeddings (recordatorio)

- Word2Vec

- Vectores que representan palabras, y guardan relación con su significado
- Generados con entrenamiento no supervisado

RNN y LSTM (recordatorio)

ELMO (recordatorio)

- Utiliza redes **LSTM bidireccionales** para extraer el embedding
  - El embedding hace mejor uso del contexto

- Una palabra no tiene representación única
  - Un embedding diferente **para una misma palabra** en oraciones diferentes

Transformers

Waswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). **Attention is all you need**. *Advances in neural information processing systems*, *30*.

Transformers

Transformers. Dos conceptos clave

- Atención
- embeddings posicionales

Atención: Vector key, vector query

- Vector key
  - Representa a la palabra actual
- Vector query
  - Representa una búsqueda de palabras que encajarían con la actual

Vector key, vector query. Vector de atención

- Vector key
  - Representa a la palabra
- Vector query
  - Representa las partes de la frase a las que esta palabra presta atención

Yo_K	[-0.33 -0.44 ... 0.11]
Voy_K	[-0.76 -0.59 ... -0.40]
A_K	[0.14 -0.15 ... 0.43]
$Aprobar_K$	[-0.08 0.01 ... 0.48]

[0,13	$Yo_Q * Yo_K$
0.87	$Yo_Q * Voy_K$
0,06	$Yo_Q * A_K$
0,52]	$Yo_Q * Aprobar_K$

Atención

[CLS]
it
looks
small
outside
,
but
quite
big
inside
nice
styling
too

embeddings posicionales

embeddings posicionales

Yo_p	[1 1 … 1]
Voy_p	[2 2 … 2]
A_p	[3 3 … 3]
$Aprovar_p$	[4 4 … 4]

Yo	[-0.33 -0.44 ... 0.11]
Voy	[-0.76 -0.59 ... -0.40]
A	[0.14 -0.15 ... 0.43]
Aprobar	[-0.08 0.01 ... 0.48]

embeddings posicionales

Yo_p	[0 0 0 …0 0 0]
Voy_p	[0 0 0 …0 0 1]
A_p	[0 0 0 …0 1 0]
$Aprobar_p$	[0 0 0 …0 1 1]

Yo	[-0.33 -0.44 ... 0.11]
Voy	[-0.76 -0.59 ... -0.40]
A	[0.14 -0.15 ... 0.43]
Aprobar	[-0.08 0.01 ... 0.48]

embeddings posicionales

Yo_p
Voy_p
A_p
$Aprobar_p$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Yo	[-0.33 -0.44 ... 0.11]
Voy	[-0.76 -0.59 ... -0.40]
A	[0.14 -0.15 ... 0.43]
Aprobar	[-0.08 0.01 ... 0.48]

embeddings posicionales. Proceso paralelizable

- Los textos se dejan de tratar como secuencias
- Las palabras se procesan en paralelo

BERT

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Modelo de Lenguaje basado en **Transformers**

- Pre-entrenado con BooksCorpus (800M words) y Wikipedia en inglés (2,500M words)
- Genera embeddings contextuales
- Entrenable para multiples tareas

BERT. Preentrenamiento

- Pre-entrenado con BooksCorpus (800M words) y Wikipedia en inglés (2,500M words)
  - Masked LM (MLM)
  - Next Sentence Prediction (NSP)

BERT demo

- Ejemplo extraído de https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

https://colab.research.google.com/drive/1ZQvuAVwA3IjybezQOXnrXMGAnMyZRuPU

BETO: Spanish BERT

- BETO is a BERT model trained on a big Spanish corpus.

Canete, J., Chaperon, G., Fuentes, R., Ho, J. H., Kang, H., & Pérez, J. (2020). **Spanish pre-trained bert model and evaluation data**. *Pml4dc at iclr*, *2020*, 1-10.

Task	BETO-cased	BETO-uncased	Best Multilingual BERT	Other results
POS	**98.97**	98.44	97.10 [2]	98.91 [6], 96.71 [3]
NER-C	**88.43**	82.67	87.38 [2]	87.18 [3]
MLDoc	95.60	**96.12**	95.70 [2]	88.75 [4]
PAWS-X	89.05	89.55	90.70 [8]	
XNLI	**82.01**	80.15	78.50 [2]	80.80 [5], 77.80 [1], 73.15 [4]

BETO: Spanish BERT

- *Spanish Wikis*: Wich include Wikipedia, Wikinews, Wikiquotes and more.
- *ParaCrawl*: Spanish portion of ParaCrawl (http://opus.nlpl.eu/ParaCrawl.php)
- *EUBookshop*: Spanish portion of EUBookshop (http://opus.nlpl.eu/EUbookshop.php)
- *MultiUN*: Spanish portion of MultiUN (http://opus.nlpl.eu/MultiUN.php)
- *OpenSubtitles*: Spanish portion of OpenSubtitles2018 (http://opus.nlpl.eu/OpenSubtitles-v2018.php)
- *DGC*: Spanish portion of DGT (http://opus.nlpl.eu/DGT.php)
- *DOGC*: Spanish portion of DOGC (http://opus.nlpl.eu/DOGC.php)
- *ECB*: Spanish portion of ECB (http://opus.nlpl.eu/ECB.php)
- *EMEA*: Spanish portion of EMEA (http://opus.nlpl.eu/EMEA.php)
- *Europarl*: Spanish portion of Europarl (http://opus.nlpl.eu/Europarl.php)
- *GlobalVoices*: Spanish portion of GlobalVoices (http://opus.nlpl.eu/GlobalVoices.php)
- *JRC*: Spanish portion of JRC (http://opus.nlpl.eu/JRC-Acquis.php)
- *News-Commentary11*: Spanish portion of NCv11 (http://opus.nlpl.eu/News-Commentary-v11.php)
- *TED*: Spanish portion of TED (http://opus.nlpl.eu/TED2013.php)
- *UN*: Spanish portion of UN (http://opus.nlpl.eu/UN.php)

Más modelos en HuggingFace

https://huggingface.co/models

HuggingFace

Librería Transformers

https://huggingface.co/docs/transformers/index

Algunos notebooks útiles

- Comparativa Word2Vec – FastText – **BERT**
- **HuggingFace Pipeline**
- **HuggingFace FineTunning**

En resumen

- Los Transformers evolucionan sobre LSTM y RNN con:
  - Atención
  - Encoding posicional
- Diferentes configuraciones de Transformers dan lugar a modelos de lenguaje neuronales **preentrenados**.
- **HuggingFace** es una comunidad que contiene múltiples modelos.

En la próxima semana

¿Cómo utilizar grandes modelos de lenguaje para pre-entrenamiento y fine-tuning?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net

Procesamiento del Lenguaje Natural

Luis de la Fuente Valentín

Tema 9 –Aplicaciones del PLN

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

En el día de hoy

¿Qué aplicación de PLN es más adecuada en cada situación?

En el día de hoy…

- Recordatorio de tecnologías clave
  - Adicional: (one/few/zero)-shot-learning

- Generación de texto
- Question Answering
- Análisis de sentimientos
- Traducción automática
- Generación automática de resúmenes
- Otras (tened a mano papel y boli)

Tecnologías clave: Word embedings

- Word2Vec

- Vectores que representan palabras, y guardan relación con su significado
- Generados con entrenamiento no supervisado

Tecnologías clave: Encoder-Decoder (seq2seq)

Transformers

- Atención
- Embeddings posicionales

https://kikaben.com/transformers-encoder-decoder/

LLMs: pre-entrenamiento vs fine-tunning

One/few/zero shot-learning

Referencia bibliográfica clave

- **Speech and Language Processing (3rd ed. draft)**

  Dan Jurafsky and James H. Martin

- https://web.stanford.edu/~jurafsky/slp3/

Aplicaciones: Autocompletado de texto

```
Las cosas podían haber sucedido...

... de cualquier otra manera y, sin embargo, sucedieron así.

... de la forma más maravillosa.
... aquel martes, aunque tuvimos que esperar dos años para que sucedieran.
... tal y como dijiste.
```

- ¿Qué tecnologías pueden realizar esta tarea?

Aplicaciones: Autocompletado de texto

- `nlp = pipeline('text-generation', model='datificate/gpt2-small-spanish')`
- `print(nlp("En 1991 Miguel Induráin ganó su primer Tour de Francia"))`

https://huggingface.co/datificate/gpt2-small-spanish

Aplicaciones: Question Answering

- Objetivo: responder a preguntas que plantea el usuario

  (asumimos que nos limitamos a preguntas sobre un texto o corpus)

  - Tipos de preguntas
    - Respuesta Si/No:
      - *¿La noche de San Juan va del 23 al 24 de junio?*
    - Respuesta cerrada
      - *¿Cuándo es la noche de San Juan?*
    - Respuesta abierta
      - ¿Cómo se celebra la noche de San Juan?

  - Tipos de estrategias
    - Information Retrieval
    - Information Extraction / Response Generation

Aplicaciones: Question Answering

- Information Retrieval

Figura 23. Esquema de un sistema de IR. Fuente: Jurafsky, 2021.

- Se devuelve el documento encontrado, tal y como se encuentra en el repositorio

Aplicaciones: Question Answering

- Information Retrieval

Figura 23. Esquema de un sistema de IR. Fuente: Jurafsky, 2021.

- Un aspecto relevante es cómo se realiza la búsqueda
  - ¿Qué técnicas podemos usar para realizar la búsqueda?

Aplicaciones: Question Answering

- Information Extraction / Response Generation

Figura 26. Sistema de IR para *factoid questions*. Fuente: Jurafsky, 2021.

Aplicaciones: Question Answering

- Information Extraction / Response Generation

```
the_context = "Perico Delgado ganó el Tour de Francia en 1988. Poco después,
en 1991, Miguel Induráin ganó su primer Tour de Francia. Después ganaría
consecutivamente los de 1992, 1993, 1994 y 1995. "

nlp = pipeline('question-answering', model='mrm8488/distill-bert-base-
spanish-wwm-cased-finetuned-spa-squad2-es')

result = nlp({
    'question': "¿Quién ganó el Tour de Francia de 1991?",
    'context': the_context
    })
print(result)
result = nlp({
    'question': "¿Quién ganó el Tour de Francia de 1988?",
    'context': the_context
    })
print(result)
result = nlp({
    'question': "¿Quién ganó el Tour de Francia de 1992?",
    'context': the_context
    })
print(result)
```

Aplicaciones: Análisis de sentimientos

- Objetivo: etiquetar un texto de acuerdo a la emoción que subyace de él
  - Tipos de análisis:
    - De polaridad
    - De emociones básicas

Aplicaciones: Análisis de sentimientos

- Análisis de polaridad
- Análisis de emociones

Aplicaciones: Análisis de sentimientos

```
nlp = pipeline('sentiment-analysis',
model='finiteautomata/beto-sentiment-analysis')
result = nlp("No he visto esta película")
print(result)
```

¿Qué tecnologías podemos usar para hacer este análisis?

Aplicaciones: traducción automática

- Objetivo: Dado un texto en un idioma, obtener un texto en otro idioma con el mismo significado

Aplicaciones: traducción automática

- Evaluación del modelo: fidelidad

  - ***character F-score*** (chrF), que compara el solape en n-gramas que hay entre la frase traducida con el modelo de MT y la traducción original.

  - **BERTScore,** con la que se obtienen los *embeddings* de las palabras de la traducción de referencia $xxJJ$ y las de la traducción obtenida $xxii$ y se comparan ambos vectores,

Aplicaciones: generación automática de resúmenes

- Objetivo: generar un texto más corto que el original y que condense el significado principal de éste.

Entrenamiento de Aplicaciones: teacher forcing

- Técnica usada sólo durante la fase de entrenamiento
- Consigue una convergencia más rápida
- La fase de entrenamiento no se realiza en las mismas condiciones que la de predicción, y podría generar mal funcionamiento

https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c

Otras aplicaciones

En la próxima semana

¿Cómo se construye un agente conversacional?

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

www.unir.net