

Minería de Textos

Resumen

La minería de textos, también conocida como minería de datos de texto, es el proceso de transformar texto no estructurado en un formato estructurado para identificar patrones significativos y nuevos conocimientos. Puede utilizar la minería de textos para analizar vastas colecciones de materiales textuales para capturar conceptos clave, tendencias y relaciones ocultas.

Índice general

I	Introducción al lenguaje	4
I.1	Textos	4
I.2	Artículos	5
I.3	Minería de datos	6
I.3.1	Ontologías	9
II	Procesamiento de lenguaje natural	10
II.1	Procesado de lenguaje natural	10
II.1.1	Definición de PLN	10
II.1.2	Desafíos del NLP	10
II.1.3	Interdisciplinariedad del NLP	11
II.2	Aplicaciones prácticas	11
II.3	Niveles del análisis lingüístico y tareas NLP	11
II.3.1	El pipeline de NLP	11
II.3.2	Análisis sintáctico	12
II.3.3	Análisis semántico	12
II.4	Breve historia del NLP	13
II.4.1	NLP antes de la era Deep Learning	13
II.4.2	NLP durante la era Deep Learning	13
III	El sentido de las palabras	14
III.1	Definiciones	14
III.2	Relaciones entre acepciones de palabras	14
III.2.1	Sinonimia y antonimia	14
III.2.2	Hipónimos e hiperónimos	15
III.2.3	Meronimia y holonimia	15
III.3	WordNet	15
III.4	Desambigüación del sentido de las palabras	15
IV	Part-of-Speech (PoS) Tagging	17
IV.1	Parts of Speech (PoS)	17
IV.2	El problema del PoS tagging	17
IV.3	PoS tagging probabilístico	18
IV.3.1	Modelos ocultos de Markov (HMM)	18
IV.3.2	Campos aleatorios condicionales (CRF)	19
IV.4	PoS tagging basado en redes neuronales	20
V	Parseo de constituyentes	21
V.1	Lenguaje formal y constituyentes	21

V.2	Gramática libre de contexto	21
V.3	Parseo libre de contexto	22
V.4	Gramáticas libres de contexto probabilísticas	23
V.5	Evaluar parsers	24
VI	Parseo de dependencias	25
VI.1	Problema del parseo de dependencias	25
VI.2	Algoritmos de parseo de dependencias	26
VI.2.1	Parseo basado en transiciones	26
VI.2.2	Parseo basado en grafos de dependencias	27
VI.3	Evaluación de parsers	27
VII	Extracción de información	28
VII.1	El problema de la extracción de información	28
VII.1.1	Extracción	28
VII.1.2	Bases de conocimientos	28
VII.2	Tarea de extracción de información	29
VII.2.1	Reconocimiento de entidades	29
VII.2.2	Reconocimiento de expresiones temporales	30
VII.2.3	Extracción de relaciones	30
VII.2.4	Detección de eventos	30
VII.2.5	Completado de plantillas	31
VIII	Clasificación de texto y minería de opiniones	32
VIII.1	Introducción	32
VIII.2	Clasificador Naive Bayes	32
VIII.3	Análisis afectivo	33
VIII.4	Análisis de emociones	33
VIII.5	Léxico de sentimientos	33
IX	Modelaje de lenguaje	34
IX.1	Problema del modelaje de lenguaje	34
IX.1.1	Modelaje de lenguaje	34
IX.1.2	Modelos de lenguaje probabilísticos	34
IX.2	Modelos lingüísticos N-gramas	35
IX.2.1	Estimación de las probabilidades del N-grama	35
IX.2.2	Evaluación de modelos de lenguaje	36
IX.2.3	Suavizado	36
IX.3	Modelos de lenguaje de redes neuronales	37
IX.3.1	Feed-Forward Neural Network	37
X	Word Embeddings	39
X.1	Semántica de vectores	39
X.1.1	Significado, similitud, relación de palabras	39
X.1.2	Palabras como vectores	40
X.1.3	Tipos de vectores de palabras y embeddings	40
X.2	Sparse word vectors	40
X.2.1	Pesando palabras en un vector	40
X.2.2	Term frequency-inverse document frequency (TF-IDF)	41

X.3	Static word embeddings	41
X.3.1	Word2vec	41
X.3.2	FastText	42
X.4	Concerning word embeddings	43
XI	Redes neuronales para procesamiento de lenguaje natural	44
XI.1	Redes neuronales	44
XI.1.1	Redes feed forward	45
XI.1.2	Redes recurrentes	46
XI.1.3	Modelo encoder-decoder	47
XI.2	Word embeddings contextualizados y modelos de lenguaje pre- entrenados	48
XI.2.1	Embeddings contextualizados	48
XI.2.2	ELMo	49
XII	Transformers y modelos grandes de lenguaje	50
XII.1	Transformers	50
XII.1.1	Arquitectura	50
XII.1.2	Autoatención	51
XII.1.3	Input embeddings	51
XII.1.4	Modelaje de lenguaje	52
XII.1.5	Modelado de lenguaje enmascarado	53
XII.1.6	Preentrenamiento y fine-tuning	54
XII.2	Modelos grandes de lenguaje (LLMs)	54
XII.2.1	Preentrenamiento de LLMs	54
XII.2.2	Ajuste fino de LLMs	54
XII.2.3	Prompting de LLMs	55
XII.2.4	Extensiones de LLMs	55

Capítulo I

Introducción al lenguaje

I.1. Textos

Los humanos nos comunicamos principalmente de forma oral, y el texto es la representación escrita de ese habla. Los animales también se comunican, usualmente sobre el estado actual o presente. El texto es una abreviación del habla, una forma de transcribirla con ciertas características:

- Es muy comprimido: el habla se produce mediante la acción de los músculos controlados por la corteza motora. Estamos limitados a emitir una cadena unidimensional de sonidos o fonemas.
- Las categorías de sonido están acotadas y corresponden a los morfemas del lenguaje. Al escribir los morfemas, ya tenemos el texto.
- El texto se transmite a través de letras, símbolos discretos.

El texto es una transcripción de una señal sonora que mantiene sus propiedades esenciales. Puede copiarse indefinidamente y es resistente al ruido.

El objetivo de la minería de textos es extraer información relevante de textos escritos. Algunos términos clave son:

- Información: Una letra es una unidad de información. Un dígito tiene menos opciones (0-9) comparado con las letras (a-z), por lo que posee menos información. Para codificar todo el abecedario se requieren al menos dos dígitos. La cantidad de información depende del número de posibles mensajes. Por ejemplo, con un dígito hay 10 posibilidades, con dos dígitos 100, y con tres dígitos 1000. En informática, se usan dos dígitos binarios (0 y 1). El logaritmo en base 2 mide la información en bits: $\log_2(3) \approx 1.58$.
- Relevancia: Algo es relevante si está relacionado con un problema específico. Por ejemplo, si buscamos una sustancia que cause diabetes en ratones, lo relevante es todo lo que se relacione con esa sustancia. La relevancia depende del problema concreto.

- Fiabilidad: Debemos trabajar con información fiable, preferiblemente basada en la realidad y datos experimentales.
- Texto humano: Tiene una estructura que viene de la secuencia continua de movimientos musculares al hablar. Además, existe una estructura jerárquica dada por la gramática y la sintaxis.

La redundancia garantiza robustez ante el ruido. El texto es redundante en varios niveles:

1. Fonemas: un fonema, como la vocal "a", tiene una duración corta (décimas de segundo). Si parte de la señal se pierde, otras repeticiones ayudan a comprenderlo.
2. Variaciones pequeñas en sonidos o escritura normalmente no afectan al significado. Por ejemplo, quitando las vocales, aún podemos entender un texto. Muchos lenguajes antiguos, como el árabe o los lenguajes semitas, no escriben vocales. Estas se introdujeron para facilitar la lectura sin conocer el contexto.

Ejercicio: estimar la velocidad de transmisión entre humanos de texto en dos contextos: uno escribe y el otro lo recibe; uno lee algo ya escrito como una novela. No se puede estimar la información, solo un límite superior a la información ("no más que x ").

- Vídeo: 5 000 000 bits por segundo
- Teléfono: 3 000 bps
- Radio: 20 000 bps
- Leer (para uno mismo, en silencio): de media 200 palabras por minuto, unas 1000 letras por minuto. Siendo 6 bits la letra, 6000 bits por minuto o 100 bps.
- Escribir un texto: 60-90 palabras por minuto, con una media de 5 letras por palabra: 360-540 letras por minuto. Siendo 6 bits la letra, sale 40-60 bps

No toda comunicación es habla humana. Por ejemplo, si una persona describe algo para que otra lo dibuje, la información transmitida será muy diferente. Tenemos diferentes tipos de señales: sonora o de letras. En términos de información, la función $f(x)$ tiene menos información que su argumento x , es decir, $f(x) < x$.

I.2. Artículos

La mayoría de la información científica se encuentra en artículos. Un artículo es un texto publicado que puede incluir imágenes y fórmulas. Los artículos científicos y periodísticos se parecen: ambos tienen autores y reflejan la opinión de éstos. El artículo es una función de la información del evento, por lo que contiene menos información que el propio evento. Los autores están limitados por sus conocimientos y la fuente de información a la que acceden, como videos, relatos, testimonios, etc. El tiempo

de escritura es también importante; el autor produce la mejor versión posible en ese momento, dentro de su contexto.

Un artículo de una página de periódico suele tener unas 300 palabras. Este valor puede obtenerse comprimiendo el contexto y el artículo.

En artículos científicos existe cierto grado de censura; no todo lo que sabe el autor se incluye. Cuanto más accesible es el artículo, mejor, por eso existen iniciativas OpenSource. Existen varios tipos de artículos científicos: investigación original, reviews (revisión de varios trabajos), propuestas metodológicas, estudios de caso (case studies, con potencia estadística limitada) y opiniones.

Un artículo cercano a la realidad contiene más información. Las fuentes más fieles incluyen historiales clínicos, diarios de laboratorio, datos estadísticos, comunicaciones entre personas sobre hechos, y datos del Internet of Things.

Actualmente, también están disponibles fuentes preprocesadas por inteligencia artificial (IA), que son modelos estadísticos del lenguaje. Reflejan la opinión prevalente, pero no su diversidad, tienen un horizonte temporal limitado, y un retraso por el tiempo de entrenamiento. Estas fuentes pueden reforzar opiniones dominantes debido a feedback positivo. Por ello, es recomendable complementar búsquedas IA con consultas directas a la web, aunque gran parte de la información en la web también ha pasado por IA.

I.3. Minería de datos

La minería de textos analiza uno o varios textos para responder preguntas como: ¿de qué trata? ¿Es fiable? ¿Tiene interés? ¿Qué información contiene y cómo describirla? ¿Cuál es la información nueva? ¿Cómo presentar la información de forma estructurada? Los formatos estructurados incluyen bases de datos relacionales, no relacionales, grafos y programas.

Ejercicio: Food Insecurity Interventions to Improve Blood Pressure - JAMA Internal Medicine. El artículo tiene información no textual, está en los gráficos. Tiene 458 participantes, que está bien en cuanto a potencia estadística.

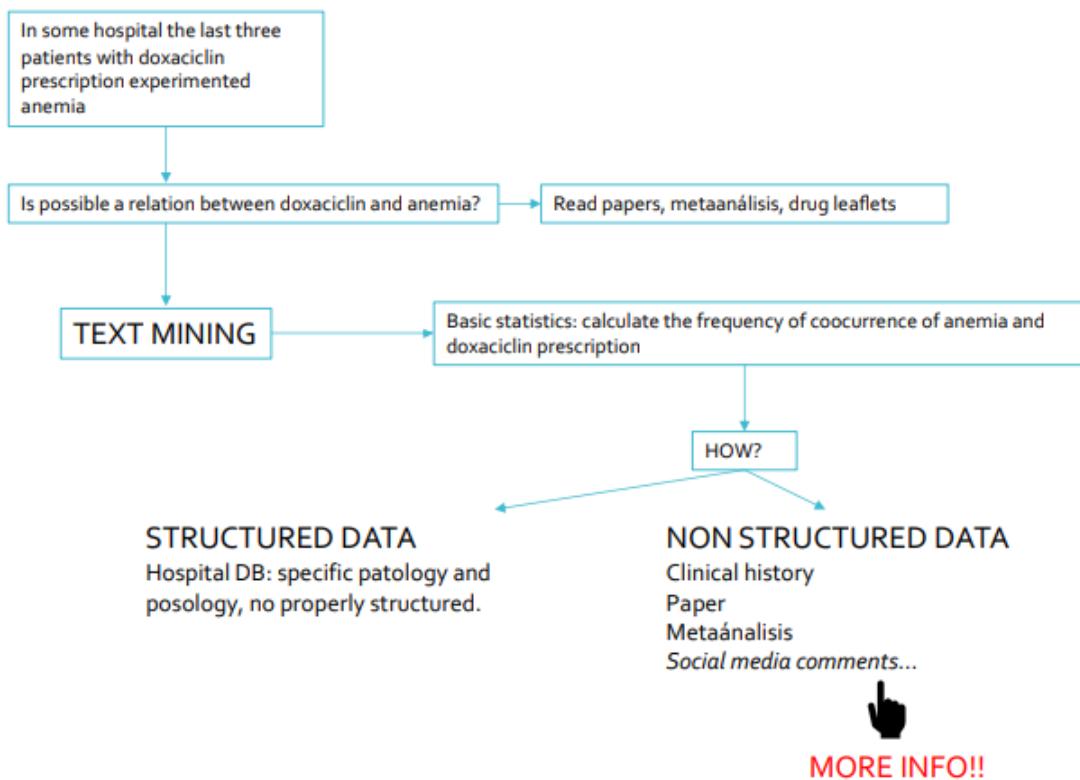
Los **datos estructurados** tienen organización clara, fáciles de consultar, modificar y analizar, y suelen ser isomórficos con tablas relacionadas. Ejemplos: bases de datos, hojas Excel, gráficos conceptuales, datos de formularios.

Los **datos no estructurados** carecen de un modelo fijo, están en formatos naturales mezclando información útil con redundante, y no se pueden organizar trivialmente en tablas. Ejemplos: texto, vídeo, imagen.

El hipertexto es texto con relaciones internas o externas. Se estima que el 80 % de los datos son no estructurados.

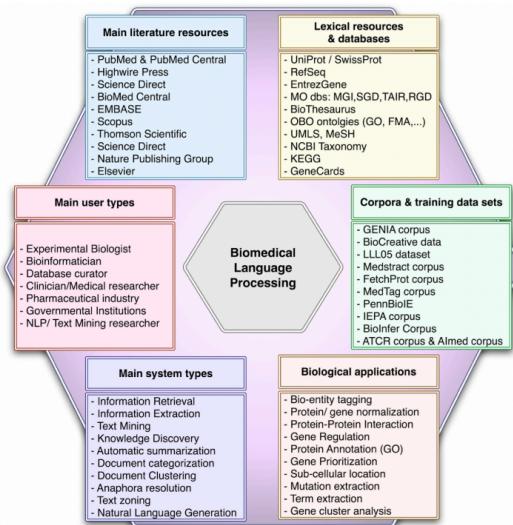
El trabajo fundamental en procesamiento de lenguaje natural es transformar datos no estructurados en estructurados. La idoneidad del método depende de la tarea real. Dos grandes grupos de métodos:

- Métodos clásicos: dividir una tarea compleja en subtareas simples (p. ej. sistema de preguntas y respuestas basado en etiquetado POS, análisis sintáctico,



reconocimiento de entidades nombradas) y combinarlas para generar la respuesta en lenguaje natural.

- Métodos basados en aprendizaje profundo: un sistema único que procesa grandes cantidades de texto para crear una representación interna útil a la tarea.

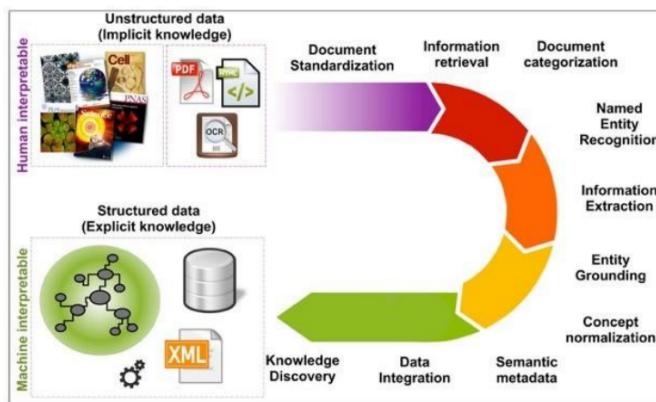


Linking genes to literature: Text mining, information extraction, and retrieval applications for biology *M. Krallinger et. al*

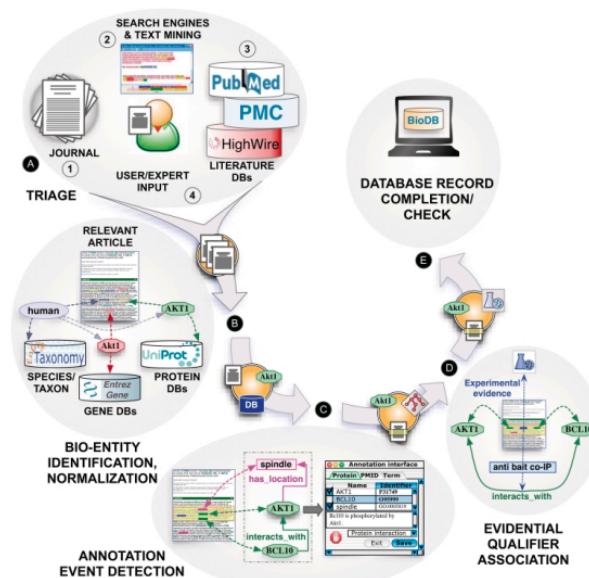
La pipeline del NLP funciona de la siguiente forma:

1. Extracción de datos de texto: Selección de las fuentes de datos de texto, aplicación de diferentes técnicas de consulta, API REST, extracción de datos web, consulta de bases de datos

2. Preprocesado: Formateo y estandarización de los datos de texto. Incluye tokenización, derivación, eliminación de palabras vacías y corrección gramatical.
3. Transformación de características: Transformación de los datos de texto en características procesables por ordenador. Codificación N-gram, etiquetado POS, análisis sintáctico, agrupación Brown, vectores de palabras.
4. Generación del modelo
5. Análisis y tratado de resultados: estandarización, almacenamiento de los resultados



La biocuración es el proceso de recogida, verificación, organización y mantenimiento de datos biológicos para asegurar su calidad, coherencia y utilidad. En el contexto de la bioinformática, la biocuración implica seleccionar información relevante, estandarizarla y anotarla correctamente en bases de datos especializadas, facilitando así su análisis y reutilización en investigaciones científicas. El objetivo principal es transformar datos brutos en recursos confiables y estructurados que permitan realizar estudios precisos y reproducibles.



Pese a la ayuda de la IA, la biocuración sigue siendo un proceso manual. Sus pasos son:

- Triaje: selección de artículos relevantes. El NLP clasifica el texto, detecta las entidades y tiene un aprendizaje no supervisado.
- Normalización e identificación de bioentidades: hay varios nombres para una misma patología, síntoma, causa, etc, por lo que se deben marcar como iguales. Por ejemplo: enfermedades vasculares hipertensivas, hipertensión e hipertensión arterial hacen alusión a lo mismo, pero con palabras distintas.
- Anotación de eventos detectados: interacciones proteína-proteína, producción génica en cuanto a localización celular, etc, extracción de relaciones en general.
- Asociación de calificadores probatorios
- Comprobación y completar las entradas de la base de datos

Extraer relaciones entre entidades es una de las tareas más difíciles en bioNLP. Es difícil porque, por un lado, es complicado extraer información significativa de la muestra para construir un modelo automático y, por otro lado, los modelos heurísticos no pueden abarcar todo el espectro de casos.

I.3.1. Ontologías

Más de 1500 bases de datos biológicas activas con datos y términos de diferentes campos y más de 200 vocabularios diferentes incluidos en recursos como UMLS: ontologías genéticas, secuencias genéticas, estructura proteica, terminología médica, taxonomías de especies, etc.

El UMLS (Unified Medical Language System) integra y distribuye terminología clave, normas de clasificación y codificación, y recursos asociados para promover la creación de sistemas y servicios de información biomédica más eficaces e interoperables, incluidos los registros médicos electrónicos.

Originalmente integró 2 millones de nombres para unos 900 000 conceptos de más de 60 familias de vocabularios biomédicos, así como 12 millones de relaciones entre estos conceptos.

- Vinculación de términos con diccionarios estándar.
- Exploración de ontologías de términos.
- Búsqueda de relaciones entre términos.
- Correspondencia entre uno y muchos de diferentes fuentes de bases de datos.

Los Medical Subject Headings (MeSH) son un vocabulario controlado y organizado jerárquicamente elaborado por la Biblioteca Nacional de Medicina. Se utiliza para indexar, catalogar y buscar información biomédica y relacionada con la salud.

Capítulo II

Procesamiento de lenguaje natural

II.1. Procesado de lenguaje natural

II.1.1. Definición de PLN

El procesamiento del lenguaje natural (NLP por sus siglas en inglés) son un conjunto de métodos para hacer el lenguaje humano accesible a los ordenadores. Así, permite una comunicación "natural" entre humanos y máquinas y mejora la comunicación entre humanos (por ejemplo por traducción).

La ciencia ficción ya mostraba desde finales de los 70 máquinas que podían hablar.

II.1.2. Desafíos del NLP

El test de Turing lo desarrolló Alan Turing en 1950 para comprobar la capacidad de una máquina a mantener una conversación como si fuera un humano. El test se basa en un humano que "chattea" y debe distinguir si está hablando con otro humano o con una máquina. Si no lo puede distinguir, se considera que la máquina ha pasado el test de Turing.

NLP es difícil porque los lenguajes humanos son ambiguos, ilimitados, diversos, difusos, más y menos articulados. Hay palabras polisémicas, el orden de las palabras afecta a su significado, expresiones hechas ("dame tu teléfono" no se refiere al dispositivo, si no al número). La ambigüedad sintáctica se puede dar por la sintaxis, referencias a pronombres, y aunque para los humanos sea fácil de detectar y forme la base de muchos chistes, a la hora de pasarlo a la máquina se convierte en un problema.

A la hora de hacer las traducciones, depende la composición de palabras. Dos palabras que por separado tienen un significado, cuando están juntas pueden tener otro.

II.1.3. Interdisciplinariedad del NLP

El NLP se basa en muchas disciplinas muy diversas. Están los **lingüistas**, que buscan analizar cómo funciona el lenguaje, los ingenieros en **machine learning** e **inteligencia artificial** para poder automatizar las tareas y crear programas complejos. En general, toda la **informática** presenta la base de los algoritmos utilizados en NLP.

Estas áreas son las que diseñan los primeros algoritmos, pero con el tiempo salen cuestiones **sociales y éticas** por sesgos sociales, como pueden ser los géneros al traducir del inglés al español (nurse se traducía sólo como enfermera, no salía la alternativa de enfermero).

II.2. Aplicaciones prácticas

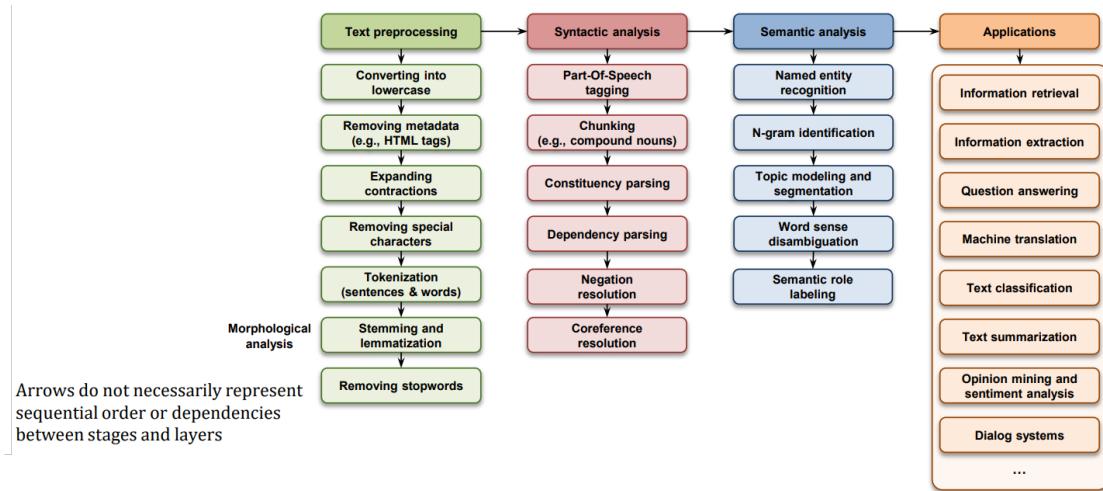
Entre las aplicaciones del NLP se encuentran:

- Traducción automática: traducción de un texto de un idioma a otro en su significado (no palabra por palabra).
- Obtención de información: dada una consulta en nuestro idioma, identifica los documentos y páginas relevantes para esa búsqueda.
- Extracción de información: se tienen documentos de texto que no están estructurados (en cuanto a información, no del texto) de los cuales se intenta extraer la información estructurada para una base de datos.
- Respuesta a preguntas: responder automáticamente preguntas propuestas por humanos en base a la información ya almacenada.
- Clasificación de texto: asigna categorías o tags a textos en base a su contenido.
- Minería de opiniones y análisis de sentimientos: identificar, extraer, cuantificar y estudiar de forma sistemática los estados afectivos y la información subjetiva (a partir del contenido textual generado por los usuarios).
- Sistema de diálogo: Un agente conversacional o chatbot es un sistema informático diseñado para conversar con un ser humano (a través de un canal de texto). Los asistentes virtuales/personales inteligentes pueden realizar tareas o prestar servicios a una persona basándose en órdenes o preguntas. Así se crearon los agentes de generación de lenguaje conversacionales como ChatGPT o Gemini. Estos algoritmos generan texto como lo haría un humano (o como se ha hecho previamente), y estos sí pasan el test de Turing.

II.3. Niveles del análisis lingüístico y tareas NLP

II.3.1. El pipeline de NLP

Para conocer un idioma, hay que manejar muchos niveles de análisis lingüístico: habla, fonética, ortografía, morfología, lexemas, sintaxis, semántica, pragmática, etc.



El primer paso es el **preprocesamiento** para normalizar todo (pasar a minúscula), quitar la metadata (tags, html), expandir las contracciones o los acrónimos, eliminar caracteres especiales, etc. También se eliminan las stopwords, que son palabras que no aportan nada de información, como los artículos o los verbos auxiliares. A continuación se stemiza y lematiza, es decir, se quitan las conjugaciones para obtener la raíz y obtener el lema (la palabra normalizada).

El siguiente paso es el **análisis sintáctico** donde se identifican las partes de la oración, se hace una composición de los grupos (nominales, preposicionales, etc) donde entra el análisis de constituyentes, resolución de dependencias y negaciones y se entra al **análisis semántico**. Con todo el procesamiento se pasan a las **aplicaciones**.

II.3.2. Análisis sintáctico

Part-Of-Speech (POS) tagging El POS tagging identifica para cada palabra o token si se trata de un nombre propio, verbo, determinante, etc.

Parseo de constituyentes Las palabras tienen su POS tag y se van agrupando las estructuras sintácticas.

Parseo de dependencias Se extrae la estructura gramática de una oración, buscando la relación entre los constituyentes definidos previamente.

II.3.3. Análisis semántico

Named Entity Recognition (NER) El reconocimiento de entidades se utiliza mucho en distintos contextos. Busca reconocer localizaciones, personas, lugares geopolíticos, números de teléfono, fechas, etc. Esto se puede utilizar posteriormente para la extracción de información.

Resolución de co-referencias Se busca qué hace referencia a la misma entidad en un texto para reemplazarlo y que el procesamiento del texto sea igual. En ocasiones, es necesario completar la información (por ejemplo, una noticia de "el presidente ha dicho ...", si a la semana hay un cambio de gobierno).

Word Sense Disambiguation (WSD) En ocasiones, las palabras pueden hacer referencia a más de una cosa por ser polisémicas o tener otras ambigüedades, por lo que se deben desambiguar. Para ello se debe ver el contexto de la oración. Generalmente, se coge la palabra de la frase con menos alternativas de ambigüedades y a partir de ella se escogen las otras acepciones.

Etiquetado semántico Al hacer el etiquetado semántico, se le está poniendo cierta taxonomía para indicar las estructuras predicado-argumento como agente, predicado, tema y localización. Son representaciones abstractas de las funciones.

II.4. Breve historia del NLP

II.4.1. NLP antes de la era Deep Learning

Los inicios se deben a lingüistas y matemáticos. En los 1960s, se formalizan las teorías gramáticas. En los 90 se generan modelos estadísticos que permitieron identificar los stopwords en base a la frecuencia. Los modelos bayesianos funcionaban muy bien.

II.4.2. NLP durante la era Deep Learning

Los primeros modelos neuronales aplicados al lenguaje entraron en el 2003 y requerían una computación enorme. En los 2010 se crearon las GPU, por lo que el tratamiento matricial mejoró con creces. También entraron los word embedding, donde se obtenían representaciones vectoriales de los conceptos vinculados a las palabras con redes neuronales. Esto marcó un antes y un después. Se crearon distintas arquitecturas con los mecanismos de atención y transformer, y en 2020 los modelos grandes de lenguaje.

Muchos de los modelos creados antes de la era Deep Learning se siguen utilizando hoy día. Si por la tarea no se requiere una arquitectura deep, se pueden utilizar los modelos "clásicos".

Capítulo III

El sentido de las palabras

III.1. Definiciones

Hay palabras que tienen varios significados. Dado un lema o una palabra sin sus flexiones (en su forma más "normal") salen cada uno de las acepciones y significados.

Un lema es una forma canónica de las palabras. Se coge la raíz y se añade lo pertinente para añadir el lema. Hay muchos algoritmos para obtenerlos, aunque son dependientes del idioma. Por ejemplo, en inglés se puede eliminar -ed o -ing para obtener el verbo sin conjugar.

Las glosas son las definiciones. Esto no le sirve al ordenador, solo a nosotros al buscar un lema en el diccionario.

Los homónimos son palabras que comparten una forma, pero tienen significados distintos. Banco puede ser la institución financiera o el asiento del parque, bat puede ser murciélagos o bate. La homonimia se identifica en la extracción de información o question-answering, y viene en función del contexto.

La polisemia es que una palabra pueda significar varias cosas relacionadas. La metonimia es la polisemia de forma sistemática. Por ejemplo, universidad puede ser el edificio o la institución.

III.2. Relaciones entre acepciones de palabras

III.2.1. Sinonimia y antonimia

El sinónimo perfecto no existe, suele haber un matiz que los diferencia, ya que si no se hubiese abandonado una de las palabras. Hay veces que el contexto no permite intercambiar dos palabras. Por ejemplo, big y large, pese a ser sinónimos, sólo es correcto big en caso de "big sister".

Los antónimos tienen significados típicamente opuestos. Ocurre similar que con los sinónimos, habiendo matices que hacen que no sean antónimos perfectos.

III.2.2. Hipónimos e hiperónimos

Los hipónimos son palabras que son más específicas que otras. Por ejemplo, coche es un hipónimo de vehículo, ya que es más específico. Hiperónimo es todo lo contrario, siendo una superclase más genérica.

III.2.3. Meronimia y holonimia

La meronimia denota una parte de algo, y holonimia lo contrario. Por ejemplo, rueda y coche, u oveja y rebaño.

III.3. WordNet

WordNet es un tesoro o diccionario, una base de datos que representa las acepciones de palabras con versiones en distintos idiomas. Representa la relación entre los significados.

Los sinsets (sets de sinónimos) son conjuntos de lemas que tienen la misma acepción y el mismo significado. Así, son equivalentes. Para una palabra polisémica se encontrará más de un sinset.

Las glosas son definiciones de texto humano que ayudan a entender el significado. Desde el punto de vista computacional interesa el sinset.

Los supersenses es una categoría de palabras donde va indicando si algo es un acto, animal, artefacto, atributo, cuerpo, etc. Son unas superclases que pueden venir bien para la desambigüación.

WordNet permite encadenar las relaciones entre palabras para crear cadenas y llegar a una clase o palabra raíz. También nos proporciona clases de tipo instancia, como suele ocurrir con las ciudades. Con esto se podría construir un grafo para extraer toda la información de un documento.

III.4. Desambigüación del sentido de las palabras

La desambiguación del sentido de las palabras (WSD) es la tarea de determinar qué sentido de una palabra se está utilizando en un contexto concreto. Tiene una larga trayectoria en la lingüística computacional y muchas aplicaciones. Busca proporcionar respuesta a preguntas como: «bat care» (¿El usuario es un vampiro? ¿O solo quiere jugar al béisbol?) o en traducción automática: traducción de «bat» como «murciélagos» (animal) o «bate» (bate de béisbol) en español. Se ha utilizado como herramienta para evaluar modelos lingüísticos.

Hay varias aproximaciones:

- Basadas en el léxico: la idea es desambiguar algunas palabras mediante aprendizaje automático por el contexto.

- Basado en todo el texto: en un lexicón con las diferentes acepciones (como Wordnet) se toma todo el texto y se busca la concordancia global, no palabra por palabra. Mapea las palabras input a los sinsets de WordNet, empezando por aquellas palabras con un solo sinset.

Hay que buscar líneas base. Algunos algoritmos cogen el **sentido más frecuente** para una palabra, escogiendo la acepción más comúnmente utilizada para cada palabra. Para WordNet esto se corresponde a la primera acepción, ya que está ordenado de mayor a menor frecuencia. Otra aproximación es un algoritmo con **un sentido por discurso**. Si todos los documentos o el discurso habla de animales, "bass" se vinculará al pez y no a la guitarra.

El **algoritmo de Lesk** se queda con la acepción en cuya glosas aparezcan palabras que también aparezcan en el texto que se quiera analizar. Esto se podría vincular también con las aproximaciones anteriores. Otros algoritmos utilizan embeddings, que son significados de palabras, y los muestra en un espacio n-dimensional. Así, se escogerá aquella acepción que se encuentre más cerca del vector de las demás palabras del texto.

La Wikipedia es una fuente de información gigantesca. Saber si una palabra está vinculada con otra, se puede utilizar la glosa de la acepción, pero quizás es demasiado corta. Por ello, se puede ir a la Wikipedia y utilizar su contenido para desambigüar.

Capítulo IV

Part-of-Speech (PoS) Tagging

IV.1. Parts of Speech (PoS)

El PoS tagging identifica las propiedades gramaticales de una palabra: nombres, verbos, adjetivos, etc. La ventaja es que hay clases cerradas (determinantes, preposiciones, pronombres, etc) y estandarizadas, aunque también hay algunas clases abiertas como nombres, verbos, adjetivos y adverbios.

Penn Treebank contiene el tagset para inglés de forma estandarizada. Tiene la nomenclatura ya establecida, asignando el tag DT a los determinantes, FW a las palabras extranjeras, IN a las preposiciones, etc.

Alternativamente está el conjunto Universal Dependencies (UD) que tiene 17 PoS tags en lugar de los 45 de Penn Treebank. Estos tags se agrupan en tres clases (abierto, cerrado y otro).

IV.2. El problema del PoS tagging

El etiquetado del PoS consiste en asignando a cada palabra o token de un texto su función dentro de la oración. A partir de ahí se podría hacer el análisis sintáctico, semántico e incluso del discurso.

Algunas de las aplicaciones son:

- Reconocimiento de entidades: nombres propios, lugar, organización, persona
- Desambigüación de palabras: diferentes significados de palabras pueden tener distintas clases.
- Traducción
- Análisis de opinión
- Preguntas a respuestas
- Sistemas de diálogos

El PoS tagging no es trivial, ya que una palabra puede tener una etiqueta distinta en función de su contexto. Esto ocurre mucho en inglés donde se adjetivan los verbos con -ing, aunque también pueden ser gerundios. La mayoría de las palabras del vocabulario no es ambigüo y se sabe exactamente lo que significa, pero el 14-15 % restante proporciona la ambigüedad al texto.

Se puede optar por un baseline y coger la más frecuente, y esto da una precisión alta, del 92 %. Otros algoritmos llegan hasta el 97 % cuando se trata del inglés con algoritmos como HMM o campos aleatorios condicionales. Entre las diferentes aproximaciones hay léxicas, basadas en reglas (si una palabra termina en -ed o -ing, puede ser un verbo), probabilísticas (HMM y CRF) y basadas en redes neuronales.

IV.3. PoS tagging probabilístico

IV.3.1. Modelos ocultos de Markov (HMM)

Las cadenas de Markov son modelos estocásticos que describen las probabilidades de secuencias de variables aleatorias (estados). La probabilidad de cada evento depende solo del estado que se ha dado en el evento anterior. La probabilidad de cada evento depende solo del estado en el que estamos y del evento.

Las cadenas ocultas tienen una serie de estados que van a venir de situaciones observables (cada uno de los tokens o palabras) y hay unos estados ocultos que son la función de cada token dentro de la oración (la etiqueta). Así, están los estados o procesos no observados (ocultos) y los procesos observables (palabras del texto). Hay dos asunciones: (1) La probabilidad de un estado concreto depende únicamente del estado anterior. (2) La probabilidad de una observación de salida depende únicamente del estado que produjo la observación y no de ningún otro estado ni de ninguna otra observación.

La matriz de transición de probabilidades representa la probabilidad de pasar de una etiqueta a otra. Si tenemos un verbo modal, lo más seguro es que a continuación venga el verbo principal, no un sustantivo. Luego se estima la maximum likelihood.

La matriz de emisión de probabilidades representa las probabilidades asociadas a cada palabra dada la etiqueta. Multiplicando las dos matrices se puede calcular la probabilidad de la etiqueta de una palabra.

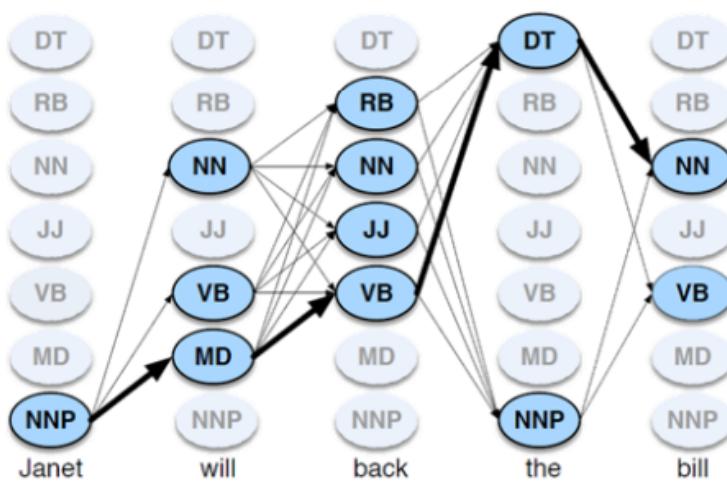
Así, el proceso tiene dos etapas:

1. Decoding: dado un input de HMM como secuencia de observaciones, se encuentran los estados más probables de la secuencia.
2. Codificación para PoS tagging: se elige la secuencia de etiquetas que van a estar vinculados a lo que se observa. Como se trata de un modelo generador, si la primera etiqueta es un nombre propio, generamos un nombre propio con cierta probabilidad. Esto no lo utilizamos nosotros, si no que aplicamos la regla de Bayes para usar las matrices de probabilidad de generación para identificar.

En definitiva, hay un texto de entrenamiento con las etiquetas ya asignadas previamente (corpus etiquetado) que se utiliza con las probabilidades para asignar las

etiquetas a un texto nuevo. Así, la probabilidad de que aparezca una palabra depende únicamente de su propia etiqueta, y es independiente de las palabras y etiquetas vecinas, y la probabilidad de una etiqueta depende únicamente de la etiqueta anterior, en lugar de toda la secuencia de etiquetas (supuesto del bigrama).

Para encontrar la secuencia de etiquetas que maximiza la probabilidad de Bayes, se utiliza el **algoritmo de Viterbi**. Es un algoritmo de programación dinámica que va buscando la secuencia más probable dada una serie de palabras. Mira todos los recorridos y luego las palabras. Coge la primera palabra y asigna su etiqueta, y busca los posibles caminos a continuación. Ve la siguiente palabra y confirma el camino correcto de las posibilidades, y calcula las siguientes ramificaciones.



IV.3.2. Campos aleatorios condicionales (CRF)

En los modelos ocultos de Markov, aprenden las etiquetas de un corpus para ahora generar etiquetas a un texto nuevo. Pero, ¿y si aparece una palabra que no estaba en el corpus? El algoritmo es fácil de implementar, pero es muy sensible cuando se altera el orden de las palabras y no haya aparecido al menos una vez en el corpus, ya que no se puede hacer la transición de los estados ocultos y no se va a reconocer como una secuencia válida. Aquí es donde entran los campos aleatorios condicionales.

Los campos aleatorios condicionales proporcionan resultados muy similares a las redes neuronales con menos maquinaria. No se utiliza un modelo generativo como en HMM, si no que se utiliza un modelo discriminativo. Se miran determinadas características de la palabra (por ejemplo, el lema y sufijos o prefijos, plurales, etc). Así, se buscan los atributos para cada palabra en una ventana de contexto específica (palabra + palabra anterior + palabra posterior, dos anteriores, dos posteriores, etc) y, en caso de que aparezca una palabra que no estaba en el corpus de entrenamiento, no pasa nada. Si acaba en -ed y tiene delante un nombre propio, con cierta probabilidad es un verbo porque se han visto patrones similares anteriormente, por lo que se etiqueta como tal. A través de las características se ve qué etiquetas no son y se discrimina así la etiqueta final. En HMM, se maximiza la probabilidad, y en CRF se discriminan todas las secuencias que maximizan la probabilidad.

Para cada palabra se define un modelo log-linear donde a cada uno de los atributos de la palabra se le asigna un peso. Aplicando unas fórmulas se llega a tener una probabilidad para la palabra. Se define una función que mapea las características de entrada a un determinado PoS tag. Se vuelve a aplicar el algoritmo de Viterbi, pero con las probabilidades calculadas a través de la CRF.

IV.4. PoS tagging basado en redes neuronales

Los CRF siguen unas estructuras similares a perceptrones. Collins consiguió en 2012 una precisión del 97.1 % con Penn Tree utilizando un perceptron¹ estructurado y las características de current word, previous words, next words, previous tag y previous two tags, similar a las características empleadas en CRF.

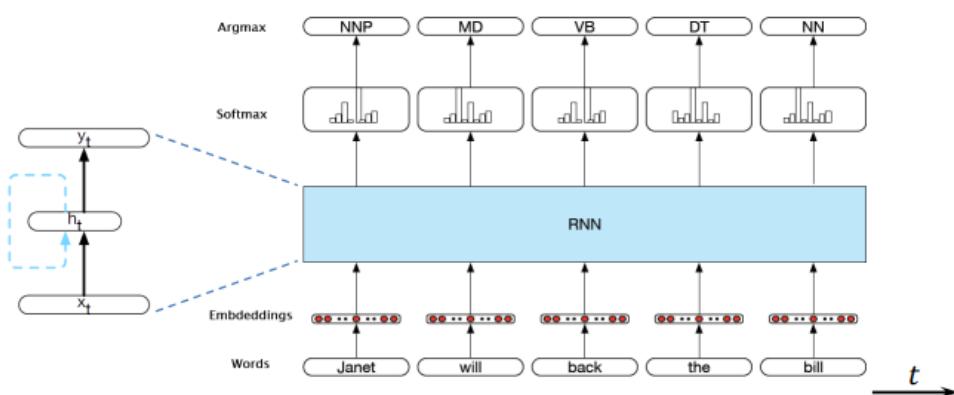


Figura IV.1: Red neuronal recurrente (RNN; recurrent neural network).

Entre una de las arquitecturas de las redes neuronales está la long short-term memory (LSTM). Es una red neuronal donde se van definiendo cómo están conectadas las neuronas y da un 95 % de precisión con CRF y 96.5 % con bi-LSTM (mira de izquierda a derecha y derecha a izquierda las palabras).

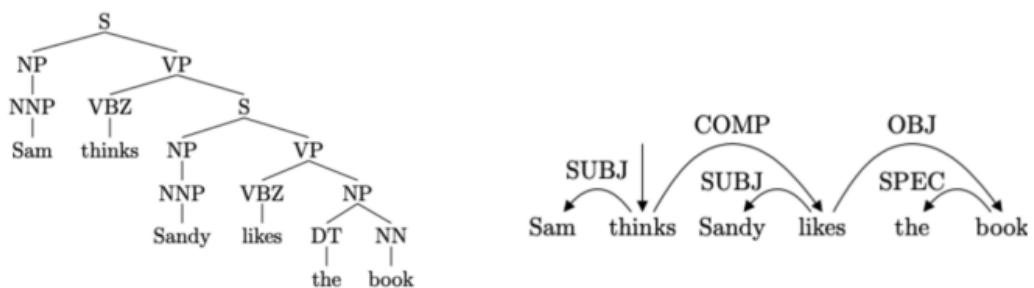
¹Un perceptron es la red neuronal más simple. Recibe un input, realiza una transformación y saca un output. Si hay varias neuronas/perceptrones, se concatena.

Capítulo V

Parseo de constituyentes

V.1. Lenguaje formal y constituyentes

Hay dos formas de parseos: el parseo de constituyentes (forma de árbol) y análisis de dependencias (saltos a partir del verbo raíz para sacar las relaciones).



La constitución sintáctica es la idea de que los grupos de palabras pueden comportarse como unidades únicas: constituyentes. Parte del desarrollo de una gramática implica crear un inventario de los constituyentes del idioma.

V.2. Gramática libre de contexto

La gramática libre de contexto o gramáticas de estructura sintáctica son el sistema formal más utilizado para modelar la estructura constituyente del inglés y otras lenguas naturales.

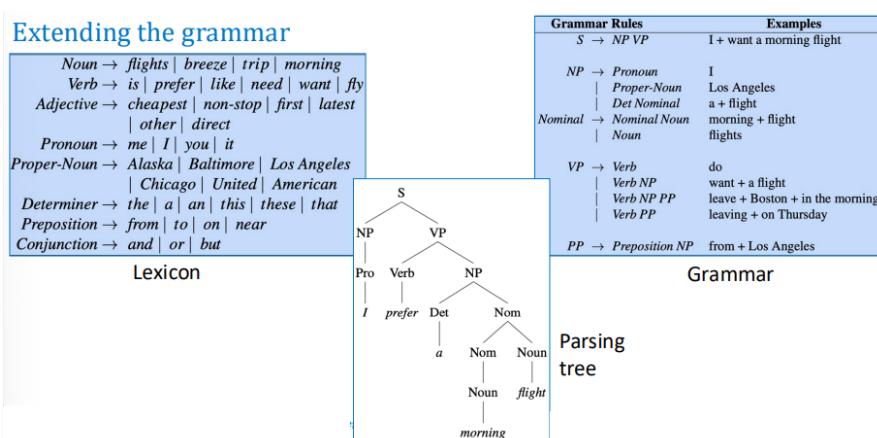
Un CFG consiste en un conjunto de reglas o producciones, cada una de las cuales expresa las formas en que los símbolos del lenguaje pueden agruparse y ordenarse, y un léxico de palabras y símbolos.

El lenguaje formal definido por una CFG es el conjunto de cadenas que se pueden derivar del símbolo inicial designado. Cada gramática debe tener un símbolo inicial designado, que a menudo se denomina *S*. Dado que las gramáticas libres de contexto se utilizan a menudo para definir oraciones, *S* se interpreta normalmente como el nodo

«oración», y el conjunto de cadenas que se pueden derivar de S es el conjunto de oraciones en una versión simplificada del inglés.

Tenemos el lexicón con opciones de nombres, verbos, adjetivos, etc. Esos son los símbolos terminales. También hay reglas de producción: la frase está compuesta por nombre y predicado. El nombre puede ser un pronombre, nombre propio o determinante nominal. El predicado verbal puede ser verbo, verbo con nombre, verbo con nombre con preposición, etc. Dado así un árbol construido con el análisis, se busca identificar si se ha podido generar con la gramática y los símbolos terminales. En esto consiste el análisis de la estructura de la oración.

Las gramáticas libres de contexto vienen definidos por los signos no terminales.



V.3. Parseo libre de contexto

El parseo es la tarea de determinar si una cadena puede derivarse de una gramática libre de contexto dada y, en caso afirmativo, cómo. La estructura del análisis sintáctico puede responder a preguntas básicas del tipo «quién hizo qué a quién». Es útil para diversas tareas posteriores, como el análisis semántico, la traducción automática y la extracción de información.

Solo es posible realizar una búsqueda exhaustiva del espacio de análisis sintácticos recurriendo a supuestos de localidad. Estos permiten realizar búsquedas eficientes reutilizando subestructuras compartidas con programación dinámica.

El algoritmo **CKY** tiene un enfoque ascendente para el análisis sintáctico en una gramática libre de contexto. Comprueba de forma eficiente si una cadena pertenece a un lenguaje, sin enumerar todos los análisis sintácticos posibles. El algoritmo primero forma pequeños constituyentes y, a continuación, intenta fusionarlos en constituyentes más grandes.

El algoritmo va realizando las agrupaciones partiendo de los símbolos terminales de la oración (lo que estamos leyendo y recibiendo) y agrupa para ir creando los constituyentes. Para que funcione bien, las gramáticas no pueden estar escritas de cualquier modo; tienen que estar escritas en CNF, un símbolo no terminal que lleva a dos símbolos no terminales o a un símbolo terminal. Esto permite así agrupar de dos en dos.

\mathcal{L}_1 Grammar	\mathcal{L}_1 in CNF
$S \rightarrow NP VP$	$S \rightarrow NP VP$
$S \rightarrow Aux NP VP$	$S \rightarrow XI VP$
$S \rightarrow VP$	$XI \rightarrow Aux NP$
 	$S \rightarrow book include prefer$
 	$S \rightarrow Verb NP$
 	$S \rightarrow X2 PP$
 	$S \rightarrow Verb PP$
 	$S \rightarrow VP PP$
 	$NP \rightarrow I she me$
$NP \rightarrow Pronoun$	$NP \rightarrow TWA Houston$
$NP \rightarrow Proper-Noun$	$NP \rightarrow Det Nominal$
$NP \rightarrow Det Nominal$	$Nominal \rightarrow book flight meal money$
$Nominal \rightarrow Noun$	$Nominal \rightarrow Nominal Noun$
$Nominal \rightarrow Nominal Noun$	$Nominal \rightarrow Nominal PP$
$Nominal \rightarrow Nominal PP$	$VP \rightarrow book include prefer$
$VP \rightarrow Verb$	$VP \rightarrow Verb NP$
$VP \rightarrow Verb NP$	$VP \rightarrow X2 PP$
$VP \rightarrow Verb NP PP$	$VP \rightarrow Verb PP$
 	$VP \rightarrow VP PP$
$VP \rightarrow Verb PP$	$PP \rightarrow Preposition NP$
$VP \rightarrow VP PP$	
$PP \rightarrow Preposition NP$	



V.4. Gramáticas libres de contexto probabilísticas

Se aplica la estadística para anotar las oraciones con treebanks y aprender modelos probabilísticos que infieran las reglas de producción.

Se pueden utilizar gramáticas suficientemente robustas, compuestas por reglas gramaticales libres de contexto, para asignar un árbol sintáctico a cualquier oración. Esto significa que es posible construir un corpus en el que cada oración de la colección se empareje con un árbol sintáctico correspondiente. Este tipo de corpus con anotaciones sintácticas se denomina banco de árboles. Los bancos de árboles desempeñan un papel importante en el análisis sintáctico, así como en las investigaciones lingüísticas de los fenómenos sintácticos.

<code>((S (NP-SBJ (DT That) (JJ cold) (, ,) (JJ empty) (NN sky)) (VP (VBD was) (ADJP-PRD (JJ full) (PP (IN of) (NP (NN fire) (CC and) (NN light))))) (..)))</code>	<code>((S (NP-SBJ The/DT flight/NN) (VP should/MD (VP arrive/VB (PP-TMP at/IN (NP eleven/CD a.m./NN)) (NP-TMP tomorrow/NN)))))</code>
(a)	(b)

(a) Brown; (b) ATIS corpora

Tenemos símbolos terminales (palabras), no terminales (constituyentes, PoS tag), reglas de producción, símbolo inicial y reglas de probabilidad.

En las gramáticas libres de contexto, se agrupan dos a dos las reglas estrictas. Ahora, se hace lo mismo teniendo en cuenta la probabilidad de las reglas de producción. En otras palabras, se añade probabilidad al algoritmo CKY con argmax.

V.5. Evaluar parsers

Para comparar los distintos enfoques de análisis sintáctico, necesitamos medir su rendimiento. Supongamos que tenemos un conjunto de análisis sintácticos de referencia (ground truth) etiquetados manualmente; por lo general, estos análisis sintácticos de referencia se extraen de un banco de árboles sintácticos como el Penn Treebank. Una solución sencilla sería la precisión por frase: el analizador se puntúa según la proporción de frases en las que el sistema y los análisis de referencia coinciden exactamente. La métrica **PARSEVAL** mide en qué medida los constituyentes del árbol de análisis hipotético se parecen a los constituyentes del análisis de referencia.

También se pueden utilizar recall y precisión. Un constituyente en un análisis hipotético C_h de una oración s se etiqueta como correcto si hay un constituyente en el análisis de referencia C_r con el mismo punto de inicio, punto final y símbolo no terminal.

$$\text{labeled recall} = \frac{\# \text{ of correct constituents in hypothesis parse of } s}{\# \text{ of correct constituents in reference parse of } s}$$

$$\text{labeled precision} = \frac{\# \text{ of correct constituents in hypothesis parse of } s}{\# \text{ of total constituents in reference parse of } s}$$

La alternativa son las agrupaciones: El número de constituyentes para los que el análisis de referencia tiene un corchete como $((A B) C)$, pero el análisis hipotético tiene un corchete como $(A (B C))$.

Capítulo VI

Parseo de dependencias

VI.1. Problema del parseo de dependencias

Una dependencia es una relación asimétrica sintáctica o semántica entre dos tokens léxicos. Un análisis de dependencias es el análisis de las relaciones entre las palabras del texto. Para ello, hay que tener un corpus etiquetado que diga que una palabra está relacionada con otra de una manera, etc.

La idea es tener la relación sintáctica o semántica entre dos tokens. Siempre habrá relaciones de la cabeza a aquello que sea dependiente y el tipo de dependencia. Las dependencias están establecidas en Universal Dependency. Así, se crea un grafo en forma de árbol que empieza en el verbo principal y añade nodos entre tokens.

Para cada palabra de la oración se intenta construir un grafo con las palabras o tokens del corpus y las relaciones de dependencia entre las palabras. La idea es tener una única cabeza (flecha que entra) a excepción de la raíz. Además, es un grafo conexo y acíclico.

El corpus que esté etiquetado tiene para cada una de las palabras la etiqueta correspondiente. Algunos idiomas tienen algunas estructuras donde se dan cruces en el grafo. Si se pueden evitar a la hora de procesar, mucho mejor.

El parseo de dependencias es una de las tareas del procesamiento de lenguaje natural donde se identifican las relaciones entre la cabeza y la palabra dependiente. El parseo es útil para traducción, extracción de información, etc, y suelen ser algoritmos rápidos.

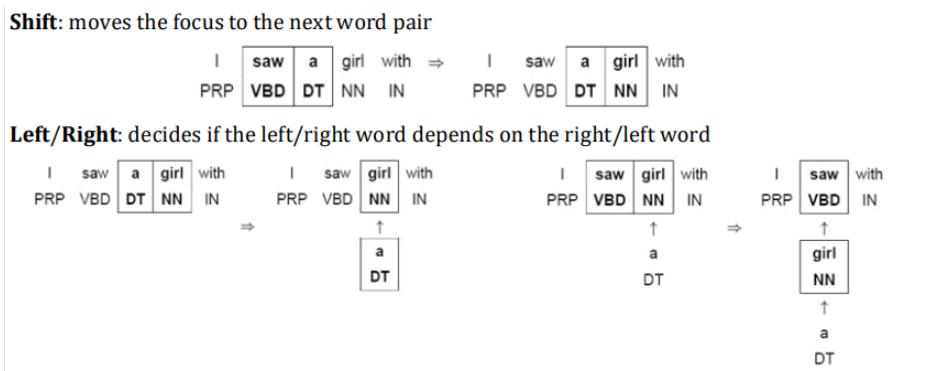
La proyectividad es que cada nodo dependiente se proyecta con un único nodo de la cabeza sin tener los cruces. Un árbol de dependencia es proyectivo si todos los arcos son proyectivos, es decir, ninguno se cruza. Si el grafo es proyectivo, será más fácil de implementar.

Existen limitaciones computacionales en las familias de algoritmos de análisis sintáctico de dependencias más utilizadas. Los enfoques basados en transiciones tienden a producir árboles proyectivos, lo que provoca algunos errores en oraciones con estructuras no proyectivas. Los enfoques de análisis sintáctico basados en grafos son más flexibles y capaces de abordar la limitación anterior.

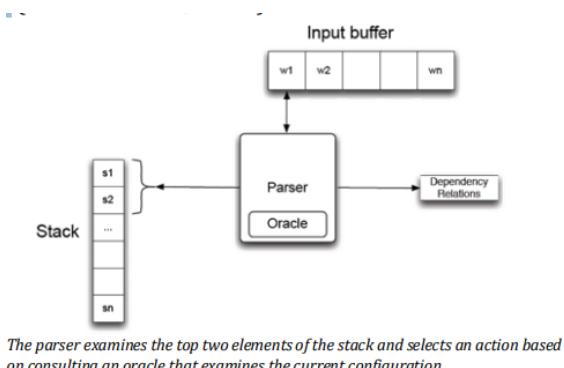
VI.2. Algoritmos de parseo de dependencias

VI.2.1. Parseo basado en transiciones

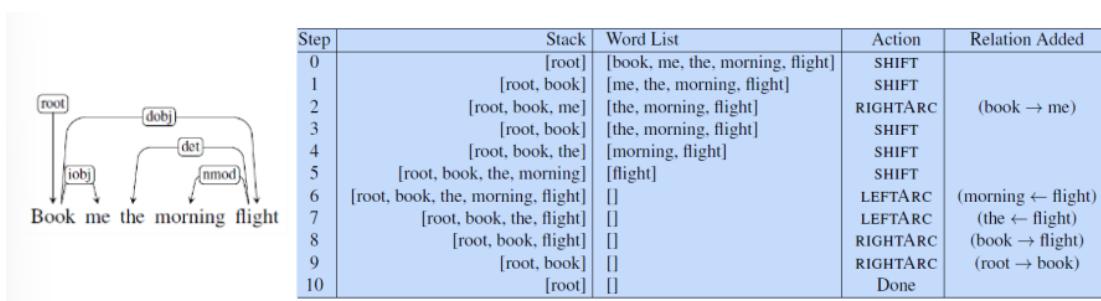
En el **parseo basado en transiciones**, hay operaciones como desplazar y reducir que buscan relaciones de dependencia entre cada pareja de palabras. Se trata de una búsqueda codiciosa que encuentra óptimos locales, siendo así mejor para dependencias locales. Si el grafo es proyectivo, tiene $O(n)$, mientras que para los no proyectivos $O(n^2)$.



Se basa en un enfoque basado en pilas denominado «análisis sintáctico shift-reduce», desarrollado originalmente para analizar lenguajes de programación. Lo complicado del parseo no es construir la pila y el buffer, si no construir el oráculo o parser para que indique si dos palabras tienen dependencia para sacarlas de la pila y guardar la dependencia.



El oráculo se crea con un algoritmo de aprendizaje automático o entrenando con un corpus. Así se van aplicando las relaciones encontradas.



VI.2.2. Parseo basado en grafos de dependencias

Los **parseos basados en grafos** construyen un grafo completo con nodos dirigidos y con peso para encontrar el score más alta. Esta búsqueda es exhaustiva y encuentra el óptimo global. No importa si el grafo es proyectivo o no, pero el coste es de $O(n^3)$.

Se busca para maximizar las relaciones de dependencia con una métrica de score oportuna para la implementación. Entre las múltiples aproximaciones, está el **algoritmo de Chu-Liu-Edmond** que parte de una serie de premisas como grafo conexo, con peso y dirigido. Se pueden crear grafos más complejos con muchas relaciones de las que hay que decidir con cuál quedarnos.

VI.3. Evaluación de parsers

Hay métricas a nivel de corpus y a nivel de oración. Hay métricas que ayudan a evaluar cuán bien funciona el parser, pero tampoco vamos a entrar en más detalle.

En resumen: tenemos el PoS tagging donde etiquetamos, el análisis de constituyentes y análisis de dependencias. Para cada uno de ellos necesitamos un corpus para entrenar y poder hacer el análisis posterior. Para cada análisis hay métricas que permiten evaluar el funcionamiento. PoS tagging es la entrada a los otros dos análisis.

Capítulo VII

Extracción de información

VII.1. El problema de la extracción de información

VII.1.1. Extracción

Tenemos muchos documentos y la intención es automatizar la construcción de una base de datos de conocimiento estructurado. Se puede trabajar con documentos sencillos o múltiples.

Las entidades son únicas, como son personas, lugares, organizaciones y fechas. Las relaciones incluyen un predicado y dos argumentos. Los eventos incluyen distintos tipos de argumentos, como por ejemplo un libro tiene un título, autor y fecha de salida.

La extracción de información no es igual a la recuperación de información. La extracción obtiene información relevante de documentos basado en un análisis lingüístico del texto. Por el contrario, la recuperación de información obtiene los documentos relevantes de una colección, utilizando generalmente modelos de bag of words.

Tras la extracción de la información, se debe guardar en alguna base de datos que permita posteriormente realizar búsquedas. Primero se suelen reconocer las entidades para posteriormente buscar la relación entre las distintas entidades.

VII.1.2. Bases de conocimientos

La DBpedia es la ontología de la Wikipedia. Incluye la información de Wikipedia estructurada en forma de pares de propiedades y valores. Básicamente incluye los infoboxes que aparecen en el recuadro de cualquier página de Wikipedia, tablas, listas, etc. No está en una base de datos per se, si no que está en formados basados en XML como RDF u OWL. Los datos RDF consisten de tripletes sujeto predicado objeto. Esto es lo que tiene la DBpedia pero a gran escala. El lenguaje de consulta es SparkQL.

WordNet también es una base de conocimiento léxica. Otros ejemplos son Yago, Freebase o Linked Open Data.

VII.2. Tarea de extracción de información

VII.2.1. Reconocimiento de entidades

El reconocimiento de entidades nombradas es un problema recurrente. La idea es poder identificar dentro de un texto las personas, lugares, fechas, organizaciones etc que aparecen en el mismo. En la mayoría de los parsers son personas, organizaciones, lugares, geopolíticos y misceláneos. Para abordar el problema de reconocer y asignar la etiqueta no se puede enfocar solo en las mayúsculas, aunque sí puede ayudar como primera aproximación. Además, hay que tener en cuenta la delimitación por conjunciones, como por ejemplo en "Hospital Ramón y Cajal". Un tipo común de ambigüedad es en personas vs organizaciones vs entidades geopolíticas y depende del contexto.

Para afinar la extracción se busca capitalización, presencia de palabras clave alrededor (como "señor" antes del nombre de una persona), etc. En algunas ocasiones se utilizan gazetteers, que son listas de nombres de personas, lugares, códigos postales, etc para poder comparar con lo que aparezca en un documento. Su ventaja es que son rápidos y fáciles de implementar en todos los idiomas, pero no es posible enumerar todos los nombres ni variantes, hay posibles ambigüedades de nombres que aparezcan en más de un listado, etc.

Algunas aproximaciones se basan en machine learning para entrenar un algoritmo que ayude a segmentar la información (tarea de etiquetado de secuencias). Entre ellos se encuentran HMM, CRF y RNN. El procedimiento estándar es IO (in out), BIO (begin in out) y BIOES (begin in out end single). Esto identifica en el texto cuándo está dentro de algo etiquetado como persona, fuera, dentro de algo etiquetado como organización, etc.

[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago] route.

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Una vez que está todo etiquetado se utilizan los algoritmos analizadores para etiquetar texto nuevo y construir la base de conocimiento. La wikificación es la identificación de personas y fechas y buscar en la wikipedia si existe para poner el vínculo. De esta forma se añade información sobre el texto. En eso consiste el texto de lincado de entidades (entity linking). En la wikificación, se crea una estructura similar a un diccionario en el cual se utilizan los nombres de las páginas como nombre de las entidades. Además, incluye las páginas de redirección con nombre alternativos y páginas de desambigüación para entidades que comparten el nombre. En el propio

texto hay muchos enlaces adicionales para poblar la base de conocimiento. Las palabras en negrita dan más información sobre el propio término. Con esto se pueden unir las distintas entidades, ya que el texto está semiestructurado. Hay algunos algoritmos de lincado automático usando redes neuronales.

VII.2.2. Reconocimiento de expresiones temporales

Generalmente hay tres tipos de expresiones temporales: fechas y horas absolutas que se pueden mapear directamente al calendario, tiempos relativos vinculados con un momento anterior o posterior (la semana pasada, el próximo martes, etc.) para los que se debe conocer el momento actual para hacer los cálculos, y duraciones (durante 15 días, etc.). Algunas palabras concretas pueden indicar expresiones temporales como los nombres de los meses o festividades (Navidad, Ramadan), nombres de momentos del día (mañana, noche, amanecer).

Las aproximaciones basadas en reglas utilizan cascadas para reconocer patrones con pequeñas reglas y expresiones regulares. Como alternativa se puede utilizar un parsing de texto BIO.

Esto nos lleva a tener que etiquetar de forma normalizada. Un estándar es TIMEX3 para etiquetar los momentos específicos en el tiempo.

VII.2.3. Extracción de relaciones

Tras reconocer las distintas entidades, el siguiente paso es encontrar las relaciones entre las mismas. Esto está muy vinculado con el RDF para establecer los enlaces entre la información.

Las relaciones están estandarizadas. La ontología Automatic Content Extraction (ACE) cuenta con 17 relaciones semánticas. Algunos datasets tienen relaciones etiquetadas manualmente para poder utilizar en el entrenamiento y test de algoritmos extractores. Un ejemplo es el TACRED dataset que tiene 106.264 ejemplos y reconoce 41 tipos de relaciones.

Para hacer esto se puede utilizar aprendizaje automático o utilizar reglas mediante expresiones regulares o patrones. Con un corpus etiquetado, se recomienda utilizar un algoritmo de aprendizaje automático al ser más preciso, aunque depende del contexto se utiliza una aproximación u otra.

VII.2.4. Detección de eventos

Los eventos son las menciones a eventos en el texto. Siempre tiene un momento particular en el que se da el evento, pero normalmente se trata de verbos que introducen los eventos o informan de lo que ocurrió.

Para clasificar los eventos se pueden etiquetar como acciones, estados, percepciones, etc. Al igual que antes se modelan mediante aprendizaje supervisado mediante el tagging BIO o clasificadores multiclas. Se puede registrar la clase, tiempo verbal y aspecto.

VII.2.5. Completado de plantillas

Una plantilla es una estructura de datos que describe eventos o situaciones recurrentes. Se compone de unos slots con forma de [nombre:valor]. El completado de la plantilla busca esos eventos y los rellena en los slots.

Como entrada se necesitan los documentos etiquetados y se crean las plantillas de forma automática con machine learning.

Capítulo VIII

Clasificación de texto y minería de opiniones

VIII.1. Introducción

La clasificación de un texto asigna categorías, conocidas como clases, a textos. Esto puede ser análisis de sentimientos, respuesta a preguntas, clasificación de diálogos, fake news, identificación de autorías, identificación del idioma, identificación de spam.

Una de las clasificaciones que se puede hacer es la minería de opinión o análisis de sentimiento para analizar las evaluaciones o emociones de personas acerca de entidades como productos, servicios, organizaciones, etc. Así se puede extraer la orientación positiva, negativa o neutra expresada hacia un objeto. Algunas palabras son indicadores muy potentes de las opiniones, como "genial" o "patético".

VIII.2. Clasificador Naive Bayes

Naive Bayes es un clasificador probabilístico. Devuelve un argumento que maximiza la clase. Este tipo de clasificadores funciona bien cuando las clases están bien separadas: opiniones polarizadas, emails spam o verídicos, etc. Para no construir una red bayesiana, basta con una versión simplificada donde sólo se tienen las palabras identificadas vinculadas a la probabilidad. Se multiplican las probabilidades. Si aparece x palabra, aumenta la probabilidad de que se esté hablando de forma positiva/negativa.

$$c_{NB} = \arg \max_{c \in C} P(c) \prod_{i \in positions} P(w_i | c)$$

$$c_{NB} = \arg \max_{c \in C} \log P(c) + \sum_{i \in positions} \log P(w_i | c)$$

Aunque la clasificación de texto Naive Bayes estándar puede funcionar bien para el análisis de sentimientos, generalmente se emplean algunos pequeños cambios que mejoran el rendimiento. En primer lugar, para la clasificación de sentimientos (y otras tareas de clasificación de texto), parece importar más si una palabra aparece o no que

su frecuencia. Por mucho que en una review digan que "x fue horrible, y fue horrible, z fue horrible", eso no lo hace más horrible. Por lo tanto, a menudo mejora el rendimiento recortar el recuento de palabras en cada documento a 1. Esto se conoce como Naive Bayes binario.

También se debe tratar la negación (me gusta esta peli vs no me gusta esta peli). Una vez que aparece una negación o partícula negativa, el resto de la frase se pone un prefijo para crear un vocabulario negativo adicional. Para ello hay que tener en cuenta si luego hay contraposiciones ("el servicio fue malo, PERO la comida estaba rica").

VIII.3. Análisis afectivo

El análisis afectivo busca identificar la emoción, sentimiento, actitud que se puede encontrar en el texto. Esto lo utilizan mucho los psicólogos.

VIII.4. Análisis de emociones

El análisis de emociones busca encontrar las emociones. Según el modelo de Ekman, hay 6 emociones: sorpresa, alegría, enfado, miedo, disgusto y tristeza. Estas son las 6 emociones básicas a partir de las cuales surgen modelos como Plutchik.

No obstante, puede que haya grados de las emociones. Se juega en 3 dimensiones: valencia, arousal y dominancia. La más interesante es arousal, que es la intensidad que la emoción provoca en el estímulo.

VIII.5. Léxico de sentimientos

Cuando no hay suficientes datos etiquetados, se pueden utilizar lexicones de sentimientos, que son palabras preanotadas con sentimientos positivos o negativos. Algunos populares son General Inquirer, LIWC, Opinion lexicon y MPQA subjectivity lexicon. NRC tiene muchos lexicones. Uno es EmoLex, que define 8 emociones básicas, pero también está VAD que asigna los valores de Valence, Arousal y Dominance a 20.000 palabras.

Capítulo IX

Modelaje de lenguaje

IX.1. Problema del modelaje de lenguaje

IX.1.1. Modelaje de lenguaje

El modelaje de lenguaje es, dado un texto, ¿cuál es la siguiente palabra que vendría después de ese texto? Una vez que tenemos un modelo de lenguaje grande, se puede añadir una capa que permita la especialización, como sequence-to-sequence. Principalmente hay dos tipos: iniciales basados en N-grama y los neuronales.

Queremos modelos que sean capaces de decidir dado un texto, probabilidades para el resto de palabras y decidir qué palabra viene a continuación. Así funciona ChatGPT: coge el texto de entrada y estima las probabilidades de la siguiente palabra. Con esa palabra, estima la siguiente, y así hasta que decide parar. Recientemente se ha incrementado, permitiendo leer documentos para aumentar el contexto y enriquecer la pregunta, pero sigue generando palabra por palabra.

Estos modelos tienen muchas aplicaciones: completar texto (búsqueda de Google), corrección de errores (Grammarly), reconocimiento de voz (Alexa, Siri), traducción automática (DeepL), OCR (optical character recognition; reconocer texto) y reconocimiento de texto escrito a mano, etc. Hoy en día estamos ya muy acostumbrados a usarlos, pero hace unos años esto era impensable: resumen de textos, respuesta a preguntas, sistemas de diálogos...

IX.1.2. Modelos de lenguaje probabilísticos

Los modelos probabilísticos computan la probabilidad de una oración dada una secuencia de palabras y computar así la probabilidad de la siguiente palabra. Estas probabilidades las aprende el modelo procesando distintos corpus o textos. Dependiendo de los documentos usados, los modelos serán unos u otros. GPT es un modelo generalista que recibió artículos de Wikipedia, foros de internet, trozos de código, etc.

Para computar la probabilidad conjunta se usa la regla de la cadena:

$$P(W) = P(w_1^n) = P(w_1, w_2, w_3, \dots, w_n) = P(w_1)P(w_2 | w_1)P(w_3 | w_1, w_2) \dots P(w_n | w_1, \dots, w_{n-1})$$

$$P(W) = \prod_i P(w_i | w_1, \dots, w_{i-1}) = \prod_i P(w_i | w_1^{i-1})$$

Para estimar la probabilidad, se puede contar y dividir, pero esto no es abordable al haber muchas posibles oraciones en un texto. Por tanto, se puede simplificar: en lugar de estimar la oración entera, solo se estima una palabra con las dos-tres palabras anteriores, no toda la frase.

A la hora de modelar, se usa la asunción de Markov: "The future behavior of a dynamical system only depends on its recent history". En un modelo de Markov de nivel-k, el siguiente estado solo depende de los estados k más recientes:

$$P(W) \approx \prod_i P(w_i | w_{i-k}, \dots, w_{i-1}) = \prod_i P(w_i | w_{i-k}^{i-1})$$

IX.2. Modelos lingüísticos N-gramas

Los N-gramas son los modelos que tienen en cuenta las N-palabras anteriores del texto. Un unígrafo solo tiene en cuenta la palabra anterior, bigrama las dos anteriores, trígrafo 3 palabras, y así sucesivamente. Para cada una de las versiones se puede definir la tarea particular.

Considerar para una palabra las 1,2,3 palabras anteriores quizás no capture las palabras más importantes de la sentencia, que es la limitación más grande de estos modelos, la no captura de las dependencias a larga distancia. Por ejemplo: "The computer which I had just put into the server room on the fifth floor crashed". El sujeto es el ordenador, pero tiene muchas palabras entre el verbo.

IX.2.1. Estimación de las probabilidades del N-grama

La estimación de las probabilidades se puede hacer simplemente contando, siguiendo la estimación de maximum likelihood (MLE). Para contar, se utiliza un símbolo de start (`<S>`) y final (`</S>`).

- **Example 1:** Bigram conditional probabilities

`<S> I am Sam </S>`
`<S> Sam I am </S>`
`<S> I do not like Mondays </S>`

$$P(I|<S>) = \frac{2}{3} = .67$$

$$P(am|I) = \frac{2}{3} = .67$$

$$P(Sam|am) = \frac{1}{2} = .5$$

$$P(</S>|Sam) = \frac{1}{2} = .5$$

$$P(Sam|<S>) = \frac{1}{3} = .33$$

$$P(I|Sam) = \frac{1}{2} = .5$$

$$P(</S>|am) = \frac{1}{2} = .5$$

$$P(do|I) = \frac{1}{3} = .33$$

...

Generalmente habrá muchos 0 porque no todas las palabras pueden ir detrás de cualquier otra. Se cuentan los bigramas y se normalizan dividiéndolos por el conteo de los unigramas de la palabra dada. Como multiplicar muchos números aumenta el coste computacional y puede dar lugar a errores, se suelen sumar los logaritmos de las probabilidades (en inglés generalmente el neperiano, pero da igual). Esto mejora la eficiencia y evita el underflow numérico.

IX.2.2. Evaluación de modelos de lenguaje

Un modelo de lenguaje tiene varias formas de evaluarse:

- **Evaluación extrínseca:** El modelo se implementa en una aplicación y se mide lo que gusta y mejora. Para traducción automática, se utiliza un traductor y se ve si el resultado es bueno.
- **Evaluación intrínseca:** Se utilizan métricas para obtener una evaluación independiente de una aplicación. La más importante es la perplexidad (perplexity). Teniendo un modelo de lenguaje que recibe un texto y genera otro, se mide cómo de cercano es al test. El *juego de Shannon* le proporciona frases incompletas y el modelo debe completarla. La medida de *perplejidad* se mide como la inversa de la probabilidad del texto de salida, de forma que cuanto mayor es la probabilidad, menor la perplejidad (y cuanto menor, mejor). Como pueden tratarse valores numéricos muy pequeños, se suele calcular como su raíz. Los valores dependen del corpus, por lo que habría que evaluar varios modelos con un mismo corpus para poder establecer qué modelo es mejor en comparación (el que obtenga un menor valor de perplejidad). Que un modelo obtenga un valor X con un corpus no significa que sea bueno ni malo, debe ser en comparación con el mismo corpus. Para experimentos piloto se suele usar la perplejidad al no ser una medida muy interpretable.

Otro problema es el de las palabras desconocidas, que no aparecen en el corpus. Su probabilidad será de 0, y genera el problema de división por 0. Esto se soluciona con un carácter especial (UNK de unknown), se va al conjunto de test, se selecciona un conjunto finito y si se encuentran palabras del training que no están se reemplazan por unknown. Con eso se estiman las probabilidades con unknown como si fuera una palabra más. En la realidad actual con modelos de redes neuronales, no se emplean palabras, sino subpalabras condensadas en prefijos, sufijos, lemas, etc.

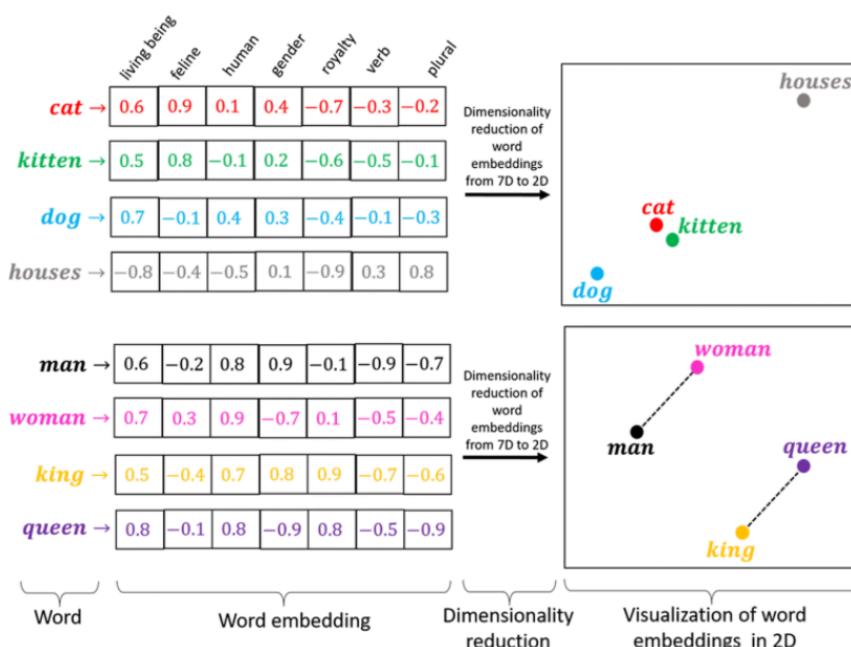
IX.2.3. Suavizado

Para las probabilidades 0 se utiliza un smoothing, un suavizado de probabilidades. Se calculan las pseudocuentas, sumando 1 a todos los N-gramas para que no haya probabilidades 0. Así, aunque se reduzca la probabilidad de N-gramas vistos en comparación con las cuentas reales, las diferencias relativas se mantienen y los cálculos se permiten.

IX.3. Modelos de lenguaje de redes neuronales

El problema de los modelos N-gramas son las dependencias a larga distancia por dependencias semánticas o sintácticas. Estas no se pueden capturar con los modelos de N-gramas, se deben usar modelos basados en redes neuronales (concretamente redes neuronales profundas).

Esencialmente, los modelos de lenguaje neuronales tienen dos componentes: la red neuronal (la arquitectura que se usa) y los embeddings. Una palabra se convierte en un embedding, un vector de números. Se trabaja así con vectores de números reales enteros. Los vectores o embeddings, si se generan bien, son componentes que tienen una dimensión. Se calculan solos, y una palabra tendrá x dimensiones o números. Se puede comprobar que capturan muchas cosas, como relaciones semánticas entre las palabras. Se puede visualizar en dos dimensiones aplicando PCA. Por ejemplo, con palabras como hombre, mujer, rey y reina, las palabras se muestran en el espacio bidimensional, estando hombre-mujer y rey-reina cercanas en el espacio. Además, la diferencia de ambos vectores será muy similar.



Estos modelos no necesitan suavizado, pueden manejar contextos más grandes, pueden generalizar mejor, tienen mayor precisión, pero son más lentos de entrenar.

Hay dos tipos principales de modelos de lenguaje neuronales: Feed-Forward Neural Network y Recurrent Neural Network.

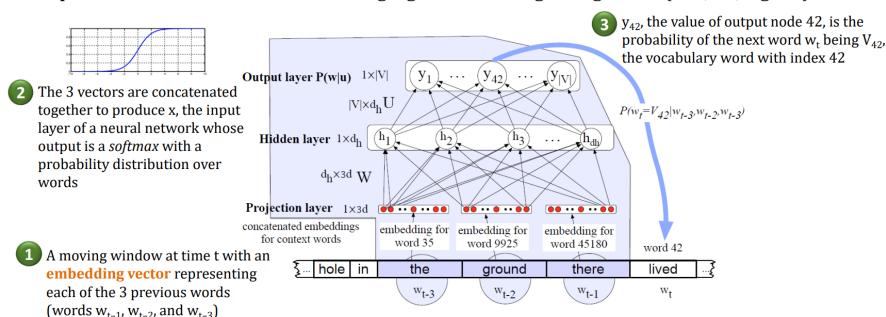
IX.3.1. Feed-Forward Neural Network

Una red neuronal tiene varias capas para realizar distintas transformaciones. Se reciben varios inputs y se transforman según la neurona para generar una salida. Inicialmente se definen capas de neuronas (los modelos profundos tienen muchas capas), como mínimo 2: una de salida y una oculta que está entre la de entrada y salida. Cada neurona tiene las entradas de la capa anterior y genera una salida que

alimenta a las neuronas de arriba. Cada arquitectura es diferente, pero esa es la base general. En el fondo, lo que ocurre es que las entradas tienen unos pesos que se deben aprender. Las neuronas tienen transformaciones que buscan aproximar una función que se desconoce. A base de proporcionar ejemplos, se van modificando los pesos y el modelo aprende. Al terminar el entrenamiento, la red queda fijada y se puede usar.

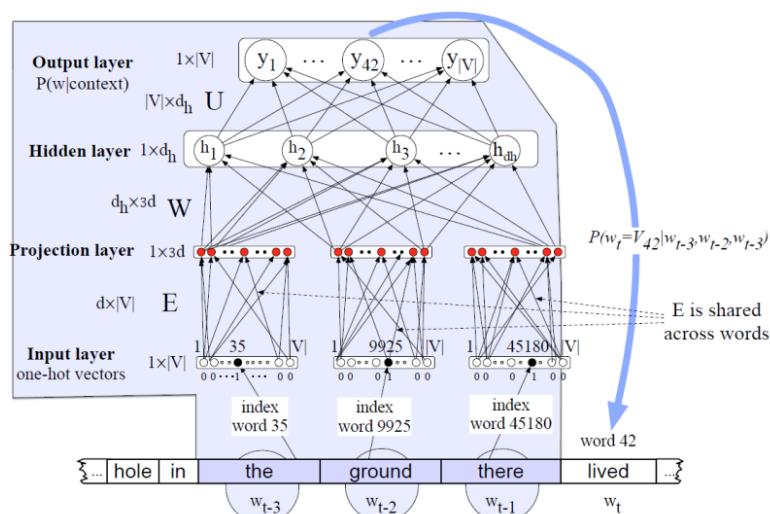
En nuestro caso, dada una secuencia, queremos predecir la siguiente palabra. Cada palabra tiene un embedding, los cuales se pueden dar a la red de neuronas al ser vectores de números. A la hora de haber establecido los pesos, al dar una entrada, la salida son unos números que indican probabilidades de las palabras del vocabulario. Cada neurona de salida es una palabra del vocabulario. Una neurona tendrá la probabilidad máxima, y será esa palabra la que se prediga.

A simplified view of a feedforward neural language model moving through a text (N=3; i.e., 4-gram)



En realidad, hay un paso intermedio anterior a los embeddings llamado one-hot. Sólo se activa la posición de la palabra en la que estamos (la palabra del vocabulario se pone en 1, el resto son 0). Las palabras generan los vectores one-hot que generan los pesos, por lo que los embeddings se convierten en los pesos.

Durante el entrenamiento, la red se inicializa de forma aleatoriamente. Se le pasan tres palabras (los modelos reales tienen unas ventanas de contexto de miles de palabras), se generan los vectores one-hot y se obtiene una salida aleatoria que será errónea. Se genera una función de pérdida/coste que mide si se ha acertado o no. Los pesos se ajustan así de arriba a abajo, modificándose por descenso por gradiente. Conforme se va entrenando el modelo, los pesos se van modificando cada vez menos al estar más ajustados.



Capítulo X

Word Embeddings

X.1. Semántica de vectores

X.1.1. Significado, similitud, relación de palabras

Previamente hemos visto WordNet, un tesauro que muestra la relación y significado de las palabras con los sinsets. Esto es de forma explícita, pero también es posible hacerlo de forma implícita con vectores semánticos.

Las palabras suelen tener pocos sinónimos, pero los significados de las palabras pueden llevar a palabras relacionadas sin que sean sinónimos. La similitud de las palabras (dicho de forma abstracta) es la que persiguen los modelos de lenguaje.

La asociación de palabras es una forma de relacionar el significado de dos palabras distinto de la similitud, como puede ser café y taza. Una forma de relación es en base al campo semántico, conjuntos de palabras que cubren un cierto dominio: hospital, cirujano, enfermero, anestesista, bisturí, etc.

Las representaciones distribuidas buscan el significado de una palabra en su uso en el lenguaje, es decir, en función del contexto en el que se usa. La hipótesis distribucional dice que las palabras que ocurren en contextos similares tienen significados similares. Para implementar esta idea, se utiliza la semántica vectorial que satisfacen lo anterior: vectores similares representarán palabras con significados similares. El aprendizaje de los vectores se obtiene en base a la distribución de las palabras en los textos, siendo así un aprendizaje auto-supervisado.

Ejemplo: ¿qué es el ongchoi? No sabemos lo que es, pero en algún momento hemos escuchado frases con ongchoi: el ongchoi está muy rico salteado con ajillo, el ongchoi está delicioso con arroz, las hojas de ongchoi se pueden usar para salsas saladas. Además, hemos visto frases con estructuras similares pero sin ongchoi: "espinacas salteadas con ajillo y arroz", "los tallos y hojas de acelgas están deliciosas", "col rizada y otras hojas saladas". De esa forma, podemos intuir que el ongchoi será un tipo de vegetal comestible. Esto emula cómo aprendemos los humanos a hablar.

Los embeddings son los vectores que asociamos a los significados de las palabras. Este concepto de embedding se suele usar para vectores densos (no hay ceros, la

dimensión suele ser más pequeña que todo el vocabulario); luego están los vectores dispersos.

X.1.2. Palabras como vectores

Los embeddings son los vectores densos que representan el significado de palabras en NLP y permiten modelar la similitud de palabras. Si los embeddings han capturado el significado correcto, las palabras se agrupan en el espacio en base a su significado: los adjetivos negativos por un lado, los positivos por otro, etc.

Para establecer la similitud se puede usar el producto escalar o normalizado por coseno del ángulo. El coseno de dos ángulos es el producto escalar dividido por el módulo de los dos vectores. El producto escalar es el multiplicando sus componentes correspondientes y sumando los resultados. El módulo es el sumatorio de los cuadrados de los componentes de un vector y tomando su raíz. El coseno puede ser entre 0 y 1 y representa lo cerca o lejos que están los vectores.

X.1.3. Tipos de vectores de palabras y embeddings

Los vectores pueden ser sparse o dense. Cuando los vectores son sparse, el vector es muy grande y tiene muchos 0. Los densos son más pequeños y no tienen 0. Dentro de los embeddings densos están los estáticos (vectores fijos para cada palabra del vocabulario) y contextualizados (vectores de una palabra son diferentes en distintos contextos).

Los embeddings densos generalizan mejor, son más fácil de usar y capturan mejor la sinonimia y otras relaciones. Por tanto, funcionan mejor.

X.2. Sparse word vectors

X.2.1. Pesando palabras en un vector

Los vectores o modelos distribucionales se basan en matrices de coocurrencia que puede ser:

- **Matriz término-documento:** primero se definió como parte del modelo vector space. Las filas son palabras, y las columnas obras de Shakespeare (por ejemplo), y en cada celda está la suma de las veces que aparece una determinada palabra en cada obra o documento. En recuperación de información, se buscan los documentos en los que aparece una palabra en base a un ranking de mayor a menor, buscando así qué documento se parece más a la consulta.
- **Matriz término-término:** tanto las filas como las columnas son el vocabulario, y se cuentan las coocurrencias de cada par de palabras en un documento con un tamaño de ventana x (por ejemplo, tres palabras a la izquierda y tres a la derecha de la palabra central a analizar). La ventana se recorre por todos los

textos. Esta forma de contar crea vectores de palabras muy grandes y muchas celdas son 0.

Estas dos matrices solo tienen las frecuencias de las palabras. No obstante, palabras demasiado frecuentes (determinantes, conjunciones, artículos) no suelen ser muy informativas sobre el contexto de una palabra. Esto da lugar a la paradoja de la frecuencia.

X.2.2. Term frequency-inverse document frequency (TF-IDF)

TF-IDF es otra forma de ponderar. El peso que se le da a un término en el documento no es solo la frecuencia del término en el documento, sino que se multiplica por el término IDF, que es la inversa del número de documentos en el que aparece el término. Así, las palabras que aparecen en todos los documentos reciben un valor de 0 (no proporcionan información sobre el contexto).

$$w_{t,d} = f_{t,d} \cdot idf_t = \log_{10}(\text{count}(t, d) + 1) \cdot \log_{10}\left(\frac{N}{df_t}\right)$$

La matriz de TF-IDF se puede usar para calcular la similitud de dos términos mediante el coseno de los ángulos de los vectores o de dos documentos con el coseno de los ángulos de los centroides de los vectores de los documentos. El vector de un documento se puede calcular de varias formas.

X.3. Static word embeddings

X.3.1. Word2vec

El embedding denso Word2vec fue un hito en NLP. Es estático, por lo que una palabra tiene siempre el mismo vector independiente del contexto. La idea que plantea es que, en lugar de contar las palabras que aparecen cerca de otra, se busca predecir sin importar los rendimientos de los clasificadores. Usa el concepto de ventana de ± 2 e intenta predecir para cada una de las palabras del contexto contra la palabra central y está o no en el contexto (clase positiva o negativa). El punto clave es asumir que las palabras van a ser vectores y que la similitud de los vectores va a permitir definir la probabilidad. Cuanto más similar sea el vector, más probable es que se encuentre en el contexto. La similitud va definida por el producto escalar (no por el coseno), simplificando así los cálculos. Se define directamente la probabilidad positiva con la sigmoide o función logística.

$$P(+) | w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})}$$

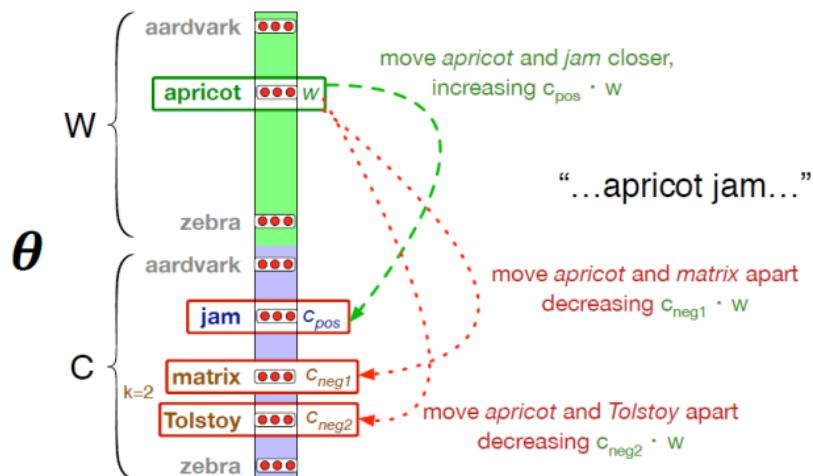
Para un contexto de x palabras, se asume que el contexto de las palabras es independiente y se puede multiplicar sus probabilidades o sumar la log-probabilidad.

$$P(+) | w, c_{1:L}) = \prod_{i=1}^L \sigma(\mathbf{c}_i \cdot \mathbf{w})$$

$$\log P(+) | w, c_{1:L}) = \sum_{i=1}^L \log \sigma(\mathbf{c}_i \cdot \mathbf{w})$$

Para la tarea de clasificación, el modelo guarda dos embeddings por cada palabra: el embedding de input (diana) y output (contexto). Así, el modelo aprende las palabras de la clase positiva y de la clase negativa (las palabras que están y que no están en el contexto). Inicialmente da vectores aleatorios, va recorriendo el texto en forma de ventana y va estimando lo bien y mal que estima una palabra con las palabras de dentro y de fuera de la ventana. Define una función de pérdida y, dada una palabra w con ejemplos positivos o negativos, el modelo asigna una probabilidad de w con un caso positivo y k -casos negativos. Busca maximizar ese producto de probabilidades, por lo que toma el menos logaritmo de dicha función. La idea es ir ajustando los pesos para minimizar el error utilizando gradientes descendentes.

A la hora de modificar los pesos, dado albaricoque, con una palabra de contexto como mermelada, se ajustan los vectores input y output para que se parezcan, pero al tener casos negativos elegidos aleatoriamente, también se busca que las palabras que no aparezcan en el contexto en las funciones input y output se alejen.



X.3.2. FastText

FastText es una adaptación de Word2vec que trabaja a nivel de subpalabra. Define n-gramas constituyentes. Para $n=3$, where se traduce a $\langle wh, whe, her, ere, re \rangle$. Estos tokens se procesan como una palabra más. $\langle \rangle$ ayuda a detectar principio y final de palabra. Hacer eso permite encontrar palabras con la misma raíz léxica (cualquier verbo conjugado) desconocidas al no estar presentes en el corpus de entrenamiento. El embedding de una palabra será la suma de los embeddings de todos los tokens que componen la palabra.

X.4. Concerning word embeddings

Los embeddings se pueden visualizar en el espacio para ver las palabras más similares, usar un algoritmo de clustering o proyectar los vectores en dos dimensiones. De esta forma se pueden ver las relaciones:

- Relación de paralelogramo: rey es a reina lo que hombre es a mujer.
- Patrones de distancias, diferencias por ser singular-plural o comparativo-superlativo.

Esto puede conllevar a que haya sesgos que vienen de los datos: si se utiliza hombre es a programador lo que mujer es a, el programa puede dar ama de casa. Otro ejemplo: father - doctor + mother = nurse. Otro sesgo es frente a ciertos grupos sociales: nombres afro-americanos muestran un mayor valor de coseno con palabras desagradables, mientras que nombres europeos-americanos tienen un coseno más alto con palabras agradables.

Capítulo XI

Redes neuronales para procesamiento de lenguaje natural

XI.1. Redes neuronales

Una red neuronal es un conjunto de neuronas conectadas que de manera general suelen dividirse en capas. Normalmente, las capas son consecutivas, aunque puede haber variantes. Dependiendo de la arquitectura, una neurona puede conectarse con todas las neuronas de la siguiente capa (full connected). Una neurona coge las entradas que recibe y multiplica cada una de ellas por su peso (combinación lineal de las entradas) y aplica una función de activación (función sigmoide, aunque también hay otras alternativas). Esa función normaliza o estandariza a unos valores acotados, para que los valores de salida estén en un rango (por ejemplo, entre 0 y 1). Para ver la salida de una neurona, depende de las transformaciones hechas en las neuronas que han producido esos valores de entrada en la capa anterior. De esta forma, se busca definir una función para la que se está entrenando la red.

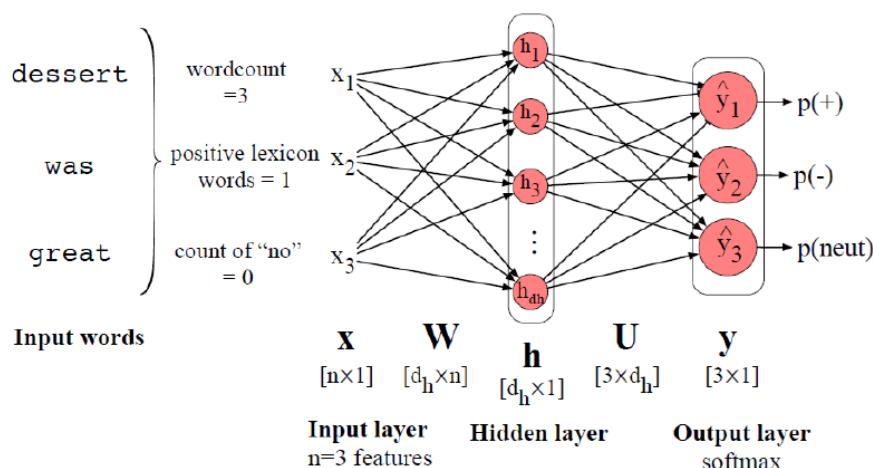
Se pueden clasificar todas las tareas de lenguaje natural en:

- **Clasificación de secuencias:** darle una etiqueta a un texto, como si tiene polaridad de opinión positiva o negativa, si un correo es spam o no, clasificar la temática de una noticia en varias categorías, etc.
- **Etiquetado de tokens (sequence labeling/tagging):** no se clasifica el texto completo, sino los tokens individuales. Por ejemplo, PoS tagging o reconocimiento de entidades nombradas. Se utiliza la estrategia BIO, BLoc, entre otras.
- **Modelado del lenguaje:** se puede utilizar para cualquiera de las demás. Dado un texto, busca estimar la siguiente palabra.
- **Sequence to sequence:** dado un texto, se genera otro. Esto es lo que utilizan los chatbots, traducción automática, resumen de un texto, etc.

Las redes neuronales permiten abordar todas estas tareas de NLP.

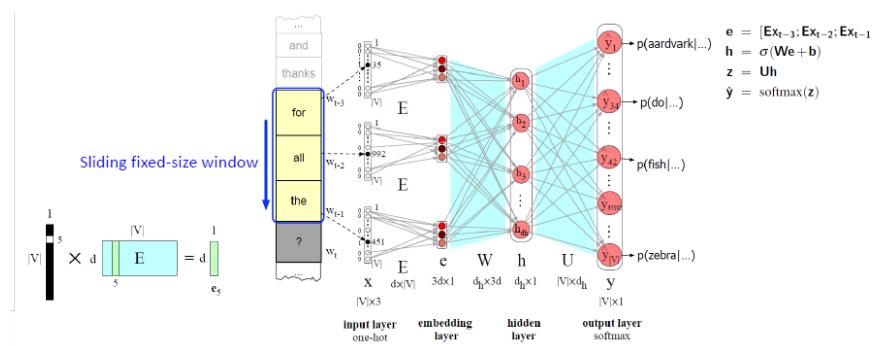
XI.1.1. Redes feed forward

Clasificación de secuencias Tenemos una red de una capa oculta y una capa de salida. En la capa de salida, cada una de las neuronas está asociada a una categoría. Por ejemplo, para el análisis de sentimiento, la capa de salida tendrá tres neuronas: una para asociaciones positivas, negativas y neutras. Todas las capas intermedias son las capas ocultas. Las conexiones tienen un vector de pesos, los cuales forman una matriz en cada paso entre capas. En este caso, la clasificación de textos tradicional (antes de los embeddings) se hacía definiendo unas características variables que describen los textos. En el ejemplo se han usado como características el número de palabras, el número de palabras positivas y el número de palabras de negación. Hay muchas características lingüísticas que se pueden elegir y procesar en los textos. La transformación puede ser sigmoide, ReLU (unidad lineal rectificada) o cualquier otra transformación no lineal.



En lugar de usar características escogidas manualmente, se pueden usar embeddings y realizar un pooling. Así, las características se aprenden de los datos.

Modelado de lenguaje En lugar de usar un embedding d-dimensional, se utiliza un vector one-hot, el embedding y la matriz de pesos interna. Además, se utiliza una ventana en el texto que se va deslizando.



Se debe definir la función de coste que permite el autoaprendizaje del entrenamiento. La función se define en base a lo que proceda, pero la opción por defecto es cross-entropy.

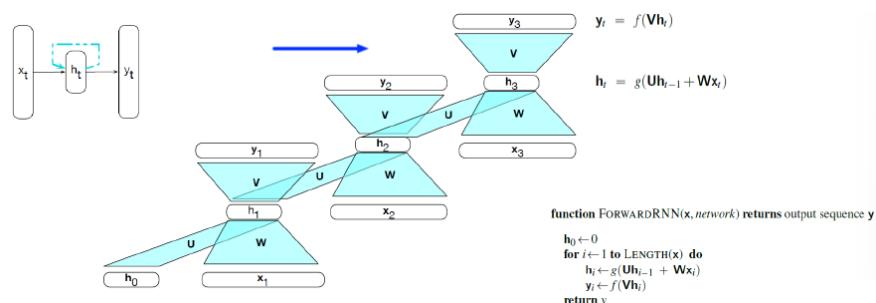
$$L_{CE}(\hat{y}, y) = -\log p(y | x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

Si la clase es 1 y predecimos (\hat{y}) 1, el error que sale es de $0 + 0 = 0$. Si son diferentes y y \hat{y} , el error que da es 1. Por tanto, esto se puede usar para ver si el modelo se confunde. Esto se puede extrapolar a muchas clases o componentes y sumar el resultado de esta función de coste.

XI.1.2. Redes recurrentes

En los modelos de N-gramas, no se consideran las largas distancias ni el contexto completo. La red neuronal feed forward sí puede capturar el contexto grande al usar sliding window, pero la naturaleza secuencial del lenguaje se pierde, por lo que aparecieron las redes neuronales recurrentes. La neurona no solo recibe la entrada de la capa anterior, sino que ahora recibe también como entrada su salida anterior (del input anterior anterior). Lo que haya pasado antes, se tiene en cuenta a la hora de los cálculos. Así se realimenta. La arquitectura general es igual, teniendo las distintas neuronas y capas, pero la neurona tiene ese añadido. El problema es que los cálculos se deben hacer de forma secuencial, por lo que estas redes son lentas. Tiene un parámetro o peso más por neurona que debe aprender la red.

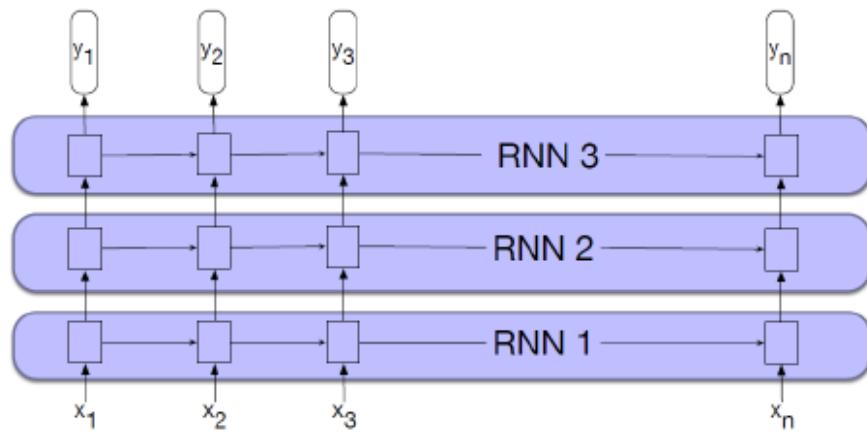
La red se representa también unrolled representando su evolución a lo largo del tiempo.



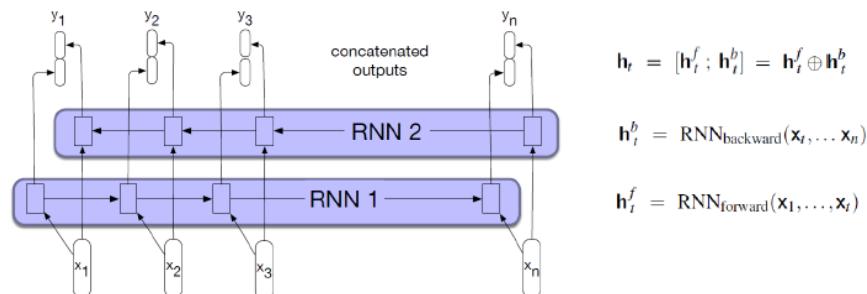
Sequence labeling Los inputs son los word embeddings, y los outputs las probabilidades de etiquetas. El hecho de tener en cuenta la salida anterior permite reforzar ciertas opciones. Por ejemplo, si la palabra anterior es un nombre propio, esto refuerza que la palabra siguiente sea un verbo.

Modelado de lenguaje Se pueden definir las probabilidades condicionales de las palabras en función de la estructura recurrente. La función de coste se define por una ventana. Cada vez que pasa un token por la red, se da un error. Al terminar, para predecir la siguiente palabra, el error tiene en cuenta los errores previos de toda la ventana (promediando los errores para los diferentes tokens). Una vez que la red está construida, se puede ir generando texto automáticamente.

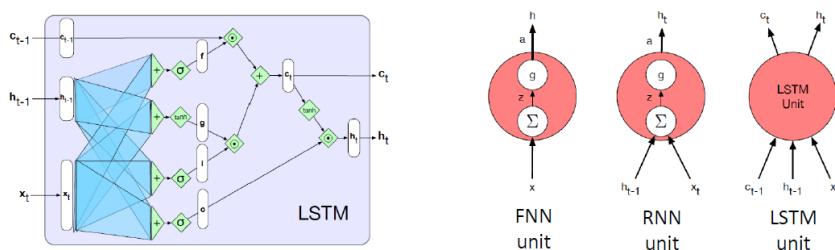
RNNs apiladas Una RNN apilada contiene muchas redes en las que el output de una capa sirve como input de otra capa. Así, unas capas pueden capturar información sintáctica y las siguientes información semántica. Si se establecen bien los parámetros, pueden mejorar, pero el tiempo de entrenamiento empieza a ser bastante elevado.



RNNs bidireccionales Una red bidireccional considera todo el input en las dos direcciones para aumentar la predicción. Se concatenan las representaciones de las dos redes recurrentes. No están en cascada las dos redes; sus salidas se concatenan.



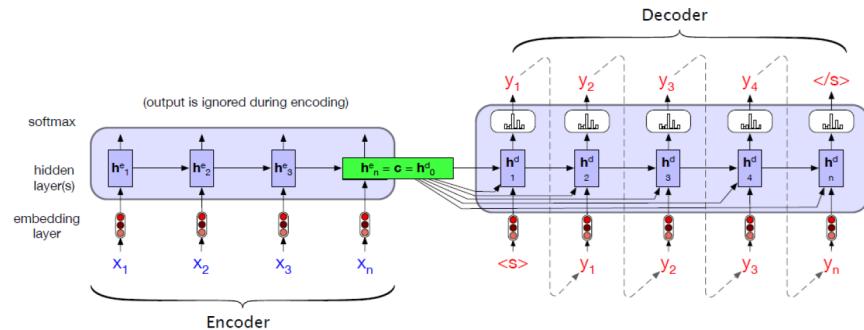
Manejo avanzado del contexto Aunque las RNN tengan acceso a toda la secuencia que precedía, se suele dar mayor relevancia a las partes más recientes. No obstante, hay casos en los que se debe retener la información distante: "The flights the airline was cancelling were full". Para ello se crearon las **LSTM** que manejan las tareas de mantener el contexto relevante a lo largo del tiempo. Aprenden a olvidar la información que ya no se necesita y recordar la información que se necesita para decisiones a futuro. Estas redes tienen unidades neuronales especializadas, por lo que su coste es aún mayor.



XI.1.3. Modelo encoder-decoder

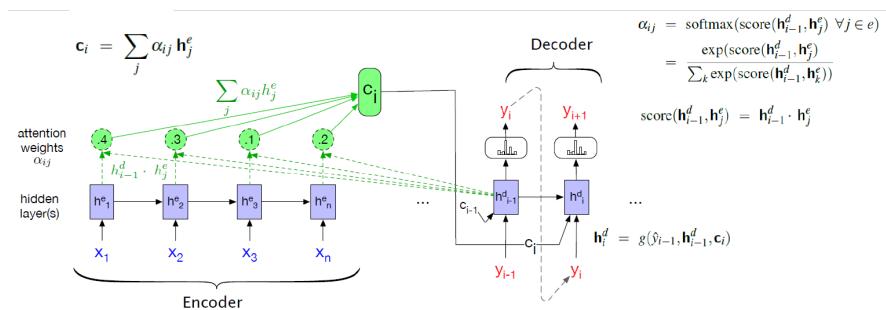
Se coge una red recurrente a la que se le van pasando entradas y genera una salida llamada contexto. Ese contexto se usa como entrada para otra red (que también

puede ser recurrente) para otra tarea. Esta estructura se llama encoder-decoder y se puede utilizar para tareas sequence-to-sequence. Así, el input primero se codifica en una representación contextualizada que se pasa al decoder para generar la secuencia output. Este modelo se puede implementar también por LSTM, no solo RNN.



Al igual que antes, se puede medir el error. Cada ejemplo de entrenamiento es una tupla de secuencias pareadas concatenadas por un token de separación. Se entrena el decodificador para la tarea, ajustando los pesos. Esto, aunque interesante, presenta un cuello de botella: hacemos que el decodificador dependa todo del estado final. Si el estado final es capaz de representar todo el input bien, no hay problema, pero si comete un error o no consigue resumir todo en un vector final, se pierden los estados intermedios. Para esto, se hace la **atención**, un punto clave de los transformer.

En el decodificador, cada estado no tiene solo en cuenta el contexto, sino que va a influir en todos los contextos previos y cada uno de ellos ponderado.



XI.2. Word embeddings contextualizados y modelos de lenguaje pre-entrenados

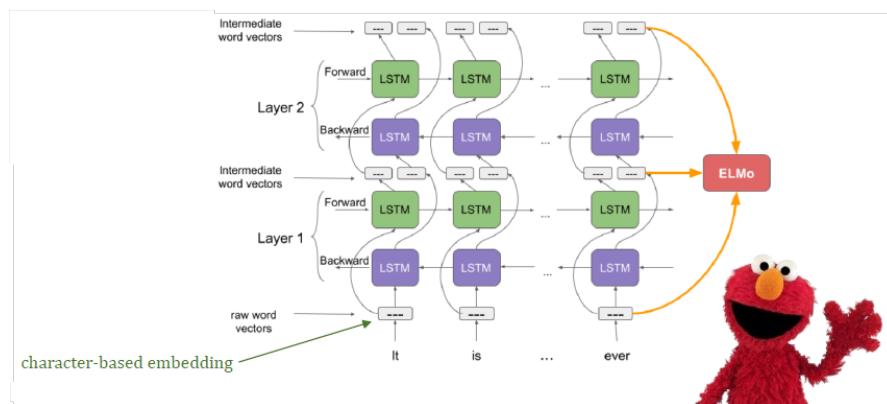
XI.2.1. Embeddings contextualizados

Los modelos estáticos asignan un embedding fijo para cada palabra, independientemente del contexto o significado. Ejemplos son Word2Vec, GloVe y FastText. Como el significado de una palabra se puede inferir de la frase en la que aparece, se puede generar un embedding contextualizado.

ELMo fue el primer modelo preentrenado que produce embeddings contextualizados. Más tarde se creó BERT, un transformer para realizar la tarea. Ambos son de código abierto y revolucionaron el área.

XI.2.2. ELMo

ELMo se basa en una red de dos capas biLSTM (una RNN apilada). Es del 2018 y representó el avance más grande en NLP. Fue el primer modelo de lenguaje que permitió la contextualización, mejorando así el rendimiento. Es un stack de LSTM bidireccionales.



La tarea para la que se entrena, al ser bidireccional, no es predecir la siguiente palabra, sino abordar la tarea de la predicción de la palabra enmascarada. De un texto, se escogen algunas palabras y se ocultan, y es el modelo el que debe inferir esas palabras que faltan o se han modificado.

Que un modelo esté preentrenado significa que el modelo ya está construido, que alguien lo ha construido con un corpus, pero que se puede especializar en una tarea concreta mediante *fine-tuning*. La más básica del ajuste fino es conectar el modelo con unas capas de lo que se quiere especializar. La definición de coste se define para las capas añadidas; las otras están congeladas y sólo se modifica lo nuevo.

Capítulo XII

Transformers y modelos grandes de lenguaje

XII.1. Transformers

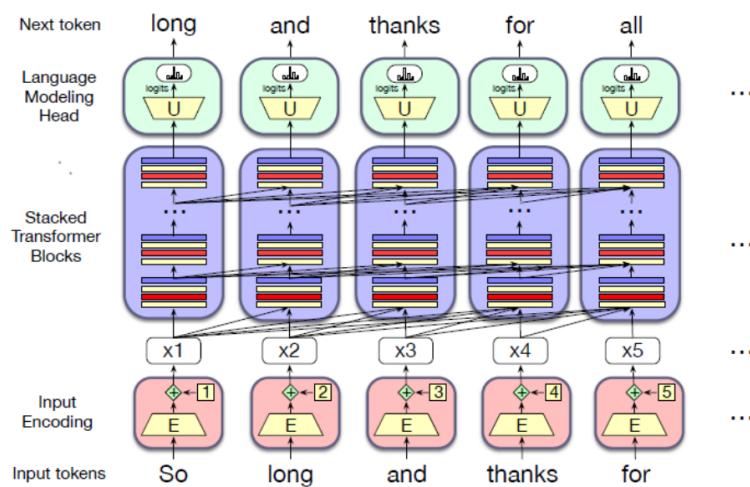
XII.1.1. Arquitectura

Un transformer es la arquitectura estándar para construir modelos grandes de lenguaje (large language models, LLMs). Se trata de una red neuronal con una estructura específica que incluye un mecanismo llamado **self-attention** (autoatención) o **multi-head attention** (atención multi-nivel).

La atención es una forma de crear representaciones contextuales del significado de un token al integrar la información de los tokens adyacentes. Así, permiten recibir contextos muy grandes. Un transformer tiene 3 partes:

- Codificación de la entrada: transformar una palabra a un embedding. Habrá varios embeddings por palabra, siendo principalmente 3: uno de la palabra en sí, uno asociado a la posición de la palabra en la oración y otro asociado a la oración en la que se encuentra.
- Stack de bloques de transformers: cada bloque es una red multicapa, con una capa de atención multi-nivel, redes feedforward y capas de normalización.
- Cabeza de modelaje de lenguaje: aprende la tarea del modelado del lenguaje. En función de la función de coste, se propaga todo el error por gradiente hacia abajo.

Bloque transformer Un bloque transformer tiene varias capas, principalmente 4: capa de autoatención, normalización, feedforward, normalización. También hay conexiones residuales para llevar la información de capas previas y que no se pierda.



XII.1.2. Autoatención

En el transformer, la atención es el mecanismo que pondera y combina las representaciones de otros tokens anteriores apropiados en el contexto desde la capa anterior para construir la representación de un token en la capa actual.

La autoatención permite al transformer extraer y utilizar directamente información de contextos arbitrariamente grandes sin necesidad de pasarlal por conexiones recurrentes intermedias como en las RNN. Al procesar cada token de la secuencia de entrada, el modelo presta atención a todos los tokens hasta el actual, incluido este. Así, las operaciones se pueden paralelizar.

La idea es la comparación de un token de interés x_i con otros tokens x_j de forma que se revele su relevancia en el contexto actual.

Los transformers permiten crear una forma más sofisticada de representar cómo las palabras pueden contribuir a la representación de entradas más largas. Tienen en cuenta tres funciones que desempeña cada incrustación de entrada durante el proceso de autoatención:

- consulta q_i = como el elemento actual en el que se centra la atención al compararlo con todas las entradas anteriores.
- clave k_i = como una entrada anterior que se compara con el foco de atención actual
- valor v_i = como un valor de un elemento anterior que se utiliza para calcular la salida del foco de atención actual

En lugar de tener una cabeza, los transformers suelen ser multi-head. Esto permite capturar relaciones sintácticas y semánticas, ya que las palabras en una oración pueden relacionarse entre ellas en distintas formas de manera simultánea.

XII.1.3. Input embeddings

Dada una secuencia de N tokens, la matriz de input de un transformer tiene la forma $[N \times d]$ y tiene un embedding para cada palabra en el contexto. El transformer

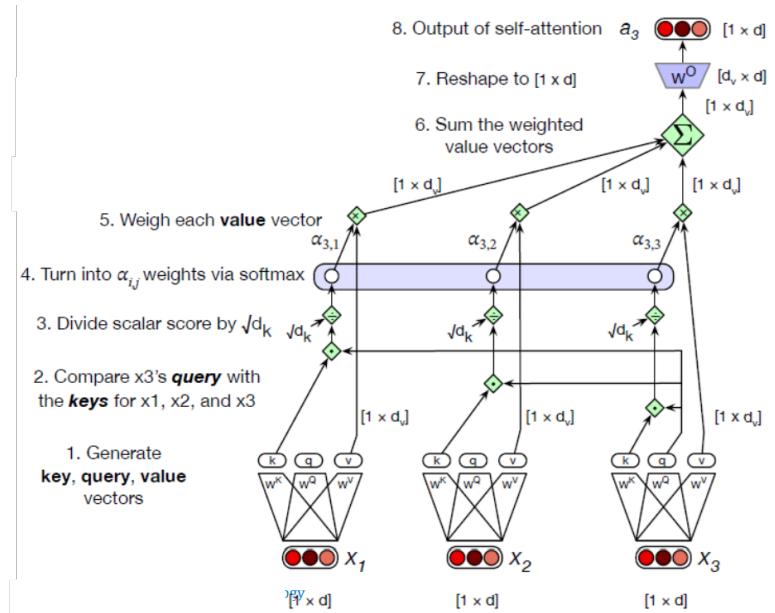
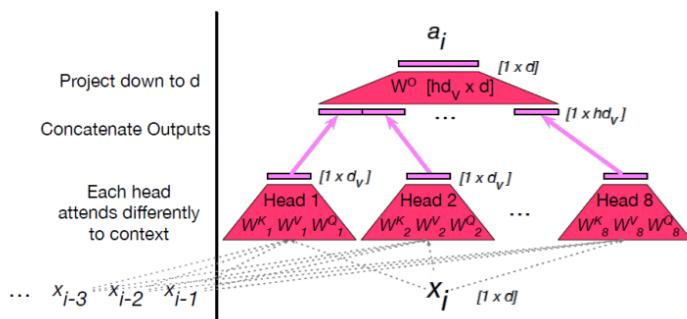


Figura XII.1: Supongamos que x_3 es la query. De hecho, es el único cuyo q va hacia arriba. Se compara con las key-words de las anteriores, se divide por d_k para normalizar. Así, la comparativa entre keywords y values es la que genera los alphas, comparando una palabra con las anteriores. Por las conexiones residuales, los values v también suben para combinar esa información contextual.

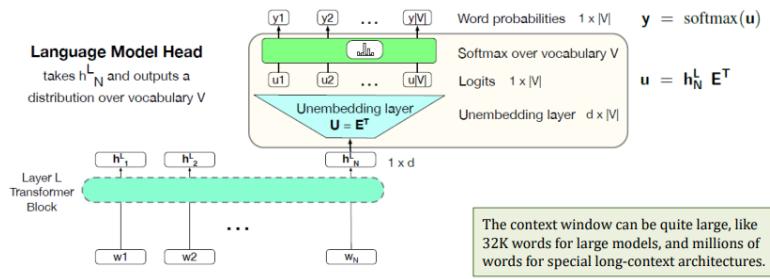


modela el orden de los tokens sin recurrencia o convolución al combinar los embeddings de las palabras de los tokens con los embeddings de las posiciones.

Los valores de embedding posicional se calculan mediante funciones sinusoidales, que ayudan a capturar las relaciones inherentes entre las posiciones (por ejemplo, la posición 4 está más estrechamente relacionada con la posición 5 que con la posición 17). Así modulan los pesos en base al orden, aprender las relaciones entre las posiciones.

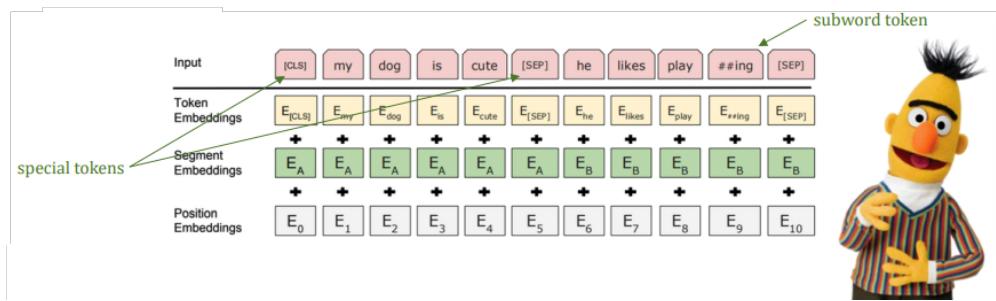
XII.1.4. Modelaje de lenguaje

La siguiente capa de modelaje es feedforward normal que coge la última capa del último bloque de transformer, el último estado, para pasarlo a la cabeza del modelado de lenguaje. Se aplican nuevos pesos para obtener la probabilidad de las palabras.



XII.1.5. Modelado de lenguaje enmascarado

La función de coste se genera en base a la tarea. En BERT se utiliza autoatención bidireccional con capas full-connected. En este caso, se busca modelar el lenguaje enmascarado, es decir, quitar o cambiar una palabra del contexto y predecir cuál era. Así, busca aprender y entender el lenguaje con la distribución de las palabras. BERT tiene dos tareas: modelaje del lenguaje enmascarado y predicción de la siguiente oración. Combina 3 embeddings: de palabra, posición y segmento.



Una vez que tenemos el modelo, se ajusta el modelo preentrenado mediante finetuning para la tarea que necesitemos y con un corpus que queramos (biomédico, etc). El modelo original de BERT tenía 768 capas ocultas, 12 capas de bloques de transformers con 12 cabezas cada una. Así, tenía 100 millones de pesos que aprender.

El modelo se presenta con una serie de oraciones del corpus de entrenamiento, donde se selecciona una muestra aleatoria (15 %) de palabras de cada oración de entrenamiento para el aprendizaje. Una palabra elegida se utiliza de una de estas tres maneras:

- Se sustituye por «[máscara]», un token de vocabulario único (80 %).
- Se sustituye por otra palabra del vocabulario, seleccionada aleatoriamente en función de las probabilidades unigramáticas (10 %).
- Se deja sin cambios (10 %).

El separador sirve no solo para aprender la palabra enmascarada, sino para predecir si dos oraciones consecutivas están realmente relacionadas o no (next sentence prediction). Así, en el corpus se le metieron 50 % de parejas de oraciones positivas y 50 % negativas o seleccionadas aleatoriamente.

Se ha visto que hasta tres capas interiores capturan información distinta para la tarea y conviene que la salida sea la media de esas últimas 4 capas del modelo.

XII.1.6. Preentrenamiento y fine-tuning

A partir de un corpus grande se construyen los modelos preentrenados. Muchos de estos modelos son públicos y se pueden descargar. Sus pesos están congelados, no se modifican, pero se pueden ajustar para crear una capa extra que sí se modifica para una tarea concreta.

En general, el ajuste fino se hace añadiendo una nueva red que se aprende para la tarea. Por ejemplo, para análisis de sentimiento, se entrena para eso. Al pasar los datos, se empiezan a generar salidas y se miden los errores, los cuales se propagan hasta el final de esa capa nueva, lo demás está congelado.

XII.2. Modelos grandes de lenguaje (LLMs)

Un LLM es un transformer en general, pudiendo ser un bloque de encoder o de decoder. Así, el encoder procesa el input y el decoder genera el output. BERT es solo un encoder, mientras que GPT es solo decoder. Otros, como T5, tienen ambos módulos. Así, un LLM puede ser encoder, decoder o encoder-decoder.

XII.2.1. Preentrenamiento de LLMs

El preentrenamiento hace referencia a la construcción de los modelos de lenguaje (normalmente basados en transformers) sobre varios corpus grandes. Durante el preentrenamiento se cogen los corpus crudos, los cuales se filtran por su calidad. Dependiendo de las clases que haya se busca quitar duplicidad de oraciones, etc. Conviene también revisar la privacidad, quitando u ocultando nombres propios, aspectos de género, etc. A continuación se tokeniza para trocear una secuencia (oraciones se dividen en palabras, lemas, etc. Cuando se tokeniza para crear un modelo, para usar el modelo, se debe usar el mismo tokenizador).

XII.2.2. Ajuste fino de LLMs

El ajuste fino es el proceso de ajustar los parámetros de un modelo preentrenado para mejorar su rendimiento en un dominio o aplicación específicos. Implica volver a entrenar el modelo preentrenado en un conjunto de datos específico, lo que permite que el modelo se adapte a una tarea específica. Por ejemplo, un modelo para generar diagnósticos médicos precisos podría ajustarse con precisión en un conjunto de datos de registros médicos, probando su rendimiento en tareas de diagnóstico médico.

El ajuste fino ayuda al modelo a especializarse en un dominio concreto, al tiempo que conserva sus capacidades generales de comprensión del lenguaje. Se trata de un tipo de aprendizaje por transferencia en el que el modelo se vuelve a entrenar con nuevos datos, con algunas o todas las capas preentrenadas configuradas para ser actualizables, lo que permite al modelo ajustar los pesos de dichas capas a una nueva tarea; los pesos de las capas restantes se mantienen «congelados» (es decir, sin cambios).

El ajuste fino eficiente de parámetros busca reducir el número de parámetros entrenables, manteniendo un buen rendimiento del modelo.

- Ajuste del adaptador: introducen módulos pequeños (adaptadores) entre las capas que se entranan mientras que los parámetros originales del modelo se quedan congelados.
- Ajuste del prefijo: se añaden vectores específicos de la tarea (prefijos) al input del modelo.
- Low Rank Adaptation: aplica modificaciones low-rank a los parámetros del modelo preentrenado.

XII.2.3. Prompting de LLMs

Uno de los aspectos principales por los que los LLM han sido tan populares y útiles es el hecho de que son accesibles para los humanos a través de conversaciones en lenguaje natural. El prompting se refiere a proporcionar entradas en lenguaje natural (consultas o indicaciones; **prompts**) a un LLM para guiar su generación de respuestas basadas en el lenguaje.

Un prompt efectivo utiliza un lenguaje claro y conciso, con contexto e información base necesaria, incluyendo instrucciones específicas o guías, incorporando inputs de ejemplos y outputs deseados y anticipando posibles desafíos.

El prompting se suele referir también como el aprendizaje in-context (ICL). Permite adaptar la capacidad del LLM a generalizar a situaciones nuevas modificando sus respuestas durante la conversaciones. Hay distintas técnicas para hacer esto:

- Zero-shot learning: utilizando solo la descripción de la tarea
- Few-shot learning: se dan algunas demostraciones o ejemplos de las salidas deseadas
- Chain-of-thought (CoT) prompting: guían el modelo por una serie de prompts interconectados, manteniendo un contexto coherente.

XII.2.4. Extensiones de LLMs

Aunque los LLM preentrenados son potentes, tienen limitaciones como alucinaciones, conocimientos obsoletos y lagunas de razonamiento. Aumentar los LLM con mecanismos externos mejora su fiabilidad, escalabilidad y adaptabilidad.

- **Generación aumentada por recuperación (RAG)**: inyectar conocimientos externos en tiempo real durante la inferencia para mejorar la precisión factual. Primero se hace un proceso de retrieval de documentos, se indexan y se añade como contexto a la pregunta.
- **Sistemas multiagente basados en LLM (MAS basados en LLM)**: uso de múltiples LLM que trabajan juntos para mejorar el razonamiento, la resolución de tareas y la toma de decisiones

Entre las ventajas de la ampliación de LLM se encuentran la mejora de la precisión y reducción de las alucinaciones, la posibilidad de ejecutar tareas dinámicas y especializadas y la mejora de la escalabilidad sin necesidad de volver a entrenar.

Retrieval-Augmented Generation RAG mejora los LLM al incorporar la obtención de conocimiento externo. La consulta suele derivar a los documentos, los cuales se procesan, trocean en chunks y esos fragmentos de los documentos crean una query extendida al incorporarse a la pregunta inicial. Esto permite proporcionar ya conocimientos de dominio específico, reduciendo así las alucinaciones y mejorar la salida. Esto se suele usar para responder a preguntas con datos a tiempo real, aplicaciones legales, médicas o científicas que requieran una precisión muy fáctica y chatbots.

Sistemas multiagentes Utilizan múltiples LLMs que interaccionan entre sí para resolver tareas complejas de forma más eficiente. Mejoran la escalabilidad y modularidad, permitiendo la especialización entre agentes y mejorando la toma de decisiones por colaboración. Se suelen utilizar como asistentes de investigación autónomos, para resolver problemas de forma colaborativa en programación y diseño y para la toma de decisiones y creación de flujos de trabajo en empresas.