Procesamiento del Lenguaje Natural

Examen Final

23 de mayo de 2022

1. **Describe brevemente las grandes fases históricas por las que ha pasado el PLN.**

   La historia del PLN puede dividirse en dos periodos: pre deep learning y post deep learning.

   El PLN tiene su origen en 1949 con el memorándum de Weaver y el objetivo de conseguir máquinas de traducción automática (MT). Durante los años siguientes surgen los primeros sistemas basados en reglas de lenguaje codificadas a mano y Chomsky introduce la idea de la gramática generativa para ayudar a las tareas MT. Sin embargo, en 1966, la ALPAC publica un informe que concluye que la MT no se puede lograr de inmediato. Entre los 60 y 70, surgen teorías basadas en gramáticas (prototipo ELIZA) y se desarrollan las ontologías conceptuales que estructuran la información del mundo en datos comprensibles por ordenadores. Durante los 80 se analiza el lenguaje utilizando enfoques simbólicos basados en reglas y gramáticas y finalmente, en los 90, los modelos estadístico revolucionan el PLN gracias a los avances computacionales y dan paso a la era post deep learning.

   En la década de los 2000, las redes neuronales comienzan a utilizarse en los modelos del lenguaje destacando los word embedding y las redes recurrentes como LSTM. Empiezan a aplicarse métodos de aprendizaje multitarea para PLN y se consigue clasificar roles semánticos y entidades con redes convolucionales. Se introduce el modelo Word2Vec capaz de captar relaciones entre palabras y se empiezan a utilizar redes neuronales recursivas inspiradas en el principio de que el lenguaje humano es inherentemente jerárquico. Transcurrida la primera época de los 2000 se empiezan a utilizar los Encoder-Decoder para el aprendizaje de secuencia a secuencia y se introduce el concepto de 'atención' en la traducción automática neuronal. Hoy en día, se utilizan modelos pre-entrenados en grandes corpus y se tunean para tareas específicas y los transformadores consiguen modelar la 'autoatención' representando palabras más sensibles al contexto

2. **Define modelo del lenguaje y argumenta algunas de sus aplicaciones en PLN.**

   Añadir que un modelo no tiene por qué ser solo probabilístico

   Un modelo de lenguaje es un modelo probabilístico calcula distribuciones de probabilidad de palabras que pertenecen a un vocabulario de dos formas principales:

   - Asignando una probabilidad a una palabra dado un conjunto de palabras anteriores ayudando a predecir qué palabra es más probable que aparezca a continuación en la oración:
     $$P(w_i|w_1, \ldots, w_{i-1})$$
   - Asignando una probabilidad a una frase completa con el objetivo de ver si una frase es más posible que otra:
     $$P(w_1, \ldots, w_k)$$

   Habitualmente, las probabilidades se estiman contando el número de ocurrencias de estos sucesos y asumiendo simplificaciones.

Entre sus aplicaciones destacan las máquinas de traducción, que utilizan el criterio de que las oraciones más probables son mejores traducciones, y el reconocimiento de escritura, basadas en la idea de que oraciones más probables son lecturas correctas. Otras aplicaciones son la corrección ortográfica, completar texto y reconocimiento de lenguaje hablado.

3. **Define modelo del lenguaje probabilístico y describe en qué consiste el modelo del lenguaje basado en N-gramas.**

   Un modelo de lenguaje probabilístico es un modelo de lenguaje que calcula la probabilidad de que una secuencia de palabras pertenezca a un lenguaje: calculando o bien la probabilidad asociada a una secuencia de palabras $P(W) = P(w_1, ..., w_n)$ o la probabilidad condicionada de una palabra dada una secuencia $P(w_n|w_1, ..., w_{n-1})$.

   Para estimar probabilidades condicionadas es buena idea aplicar simplificaciones para reducir el número de posibilidades. En concreto los modelos de Markov asumen que para un suceso existe una dependencia mayor de los sucesos recientes. Esto será clave en modelos probabilísticos como las redes neuronales o los N-Gramas.

   Un modelo del lenguaje basado en N-gramas es un modelo de lenguaje probabilístico que estima la probabilidad de cada palabra dado un contexto priori de solo $N-1$ palabras (asume Markov de orden $N-1$). Dependiendo del número de palabras priori $N-1$ se clasifican en unigramas (sin contexto), bigramas (una palabra como contexto), trigramas (dos palabras como contexto), etc. En general, el modelo de N gramas viene dado por

$$P(\mathbf{W}) = P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-N+1}^{i-1})$$

   Aunque las oraciones pueden tener dependencias a larga distancia este tipo de modelos son computacionalmente factibles y han logrado un buen desempeño en muchas aplicaciones.

4. **Describe las distintas relaciones entre significados de palabras. ¿Cuáles de ellas están incluidas y cómo se representan en WordNet?**

   Existen multitud de tipos de relaciones entre significados de palabras entre las que destacamos:

   - **Relaciones de similitud: sinónimos y antónimos.** Dos significados de palabras con distintos lema son sinónimos si presentan significados idénticos o casi idénticos. En cambio, son antónimos si presentan significados opuestos. A nivel de word embedding, los sinónimos idealmente estarán más cerca entre sí en el espacio de embedding que una palabra y su antónimo.
   - **Relaciones de taxonomía: hiponimia e hiperonimia.** El significado de una palabra es hipónimo de otro si es una subclase o instancia del otro. Por su parte, el significado de una palabra es un hiperónimo de otro si es más genérico y es una superclase del otro.
   - **Relaciones parte-todo: Meronimia y holonimia.** La relación semántica entre un holónimo y sus merónimos es como la de un todo con sus partes. De modo que, el significado de una palabra es un merónimo de otro significado si se trata de una parte del significado de la otra palabra. Por su parte, el significado de una palabra es un holónimo de otra si se trata de un todo que está integrado por partes, conteniendo a la otra palabra.

WordNet es una gran base de datos léxica del inglés que implementa todas las relaciones anteriores. Las relaciones semánticas se dan entre synset (conjunto de casi-sinónimos del significado de una palabra), no entre palabras.

La función `syn.lemmas()` obtiene los sinónimos de un synset `syn`. Por su parte, la función `l.antonyms()` encuentra los antónimos de cada lema `l`.

Las relaciones de taxonomía y parte-todo se representan mediante árboles jerárquicos. La función `t.part_meronyms()` encuentra los merónimos y la función `t.substance_meronyms()` encuentra los hipónimos dado un arbol jerárquico `t`. Finalmente, la relación de taxonomía se realiza llamando a las funciones `syn.hypernyms()` y `syn.hyponyms()` de un synset `syn` y devuelve un árbol jerárquico.

5. **Describe en qué consiste la "hipótesis distribucional"de las palabras y su aplicación para representar significados de palabra.**

   Nos encontramos en el contexto de dar un significado a una palabra representada como un vector. Queremos buscar relaciones de similaridad entre palabras.

   La hipótesis distribucional dice que "Las palabras que aparecen en contextos similares tienden a tener significados similares".

   Esta hipótesis es implementada mediante la semántica de vector, que aprende representaciones en espacios multidimensionales de significados de palabras (embeddings) a partir de la distribución en los textos de palabras vecinas, dando lugar a una tarea de aprendizaje autosupervisado. En estos espacios multidimensionales (usualmente $\mathbb{R}^k$), podemos utilizar métricas matemáticas para medir similitud entre las palabras, como la distancia del coseno. Además, usando estos embeddings, podemos utilizar técnicas de aprendizaje automático (clustering, clasificación) sobre los puntos del espacio.

6. **Describe en qué consiste las "semánticas de vector"(vector semantics), sus objetivos y los tipos de embeddings que existen.**

   La semántica de vector implementa la hipótesis distribucional mediante el aprendizaje de representaciones de significados de palabras (embeddings). Su principal objetivo es representar una palabra como un vector en un espacio semántico multidimensional derivado de las distribuciones de las palabras vecinas, dando lugar a una tarea de aprendizaje autosupervisado. De este modo, pasamos de trabajar con palabras en un lenguaje a trabajar con vectores en espacios $\mathbb{R}^k$, permitiéndonos utilizar técnicas de aprendizaje automático clásicas.

   Los dos tipos de embedding que existen son:

   - **Vectores dispersos:** el significado de una palabra se define mediante una función simple de los recuentos de las palabras cercanas (co-ocurrentes). Se trata de vectores largos y dispersos. Destaca TF-IDF y PPMI.
   - **Embeddings densos:** La representación de una palabra se crea entrenando un clasificador para distinguir las palabras cercanas y las lejanas. Se trata de vectores cortos y densos. Destacan word2vec (embedding estático) y BERT (embedding contextualizado). Usar embeddings densos tiene ventajas en los algoritmos de ML al ser de menor tamaño y, puesto que las palabras relacionadas obtienen embeddings cercanos, estos vectores pueden generalizar mejor y capturar mejor las relaciones de sinonimia y otras.

7. **Describe el funcionamiento de Word2Vec y el Skip-Gram model.**

Word2vec es un modelo sencillo que plantea una **tarea de clasificación binaria** entrenando un modelo de regresión logística. Trata de predecir cómo de probable es que dos palabras aparezcan juntas y utiliza los pesos aprendidos como embeddings, sin que importe el resultado de la tarea de predicción. Además, no se necesita que los datos estén etiquetados manualmente (self-supervised).

Word2vec utiliza el modelo **skip-gram** mediante un muestreo de ejemplos negativos y plantea la tarea de clasificación del siguiente modo:

Dada una tupla (w, c) que contiene una palabra objetivo w y una candidata a palabra contextual c (por ejemplo, (fresa, mermelada) o (fresa, conejo)), se calcula la probabilidad de que c sea realmente una palabra de contexto $P(+ \mid w, c)$ (en el ejemplo, la primera sería positiva y la segunda falsa).

En este contexto, el *modelo skip-gram* trata de hallar esta probabilidad utilizando una medida de **similaridad entre los embeddings**, utilizando habitualmente la función sigmoidal del producto escalar entre los vectores:

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

Por tanto, Word2Vec recibe un corpus como entrada, asigna un embedding aleatorio a cada palabra e iterativamente intenta mejorar los embeddings utilizando el problema de clasificación binaria anterior, maximizando la similaridad a los pares de ejemplos positivos y minimizando la similaridad con los pares de ejemplos negativos.

8. **Define la tarea de POS tagging, da ejemplos de etiquetas POS, y compara los porcentajes de acierto alcanzados por diferentes aproximaciones.**

   POS tagging es el proceso de asignar una parte del discurso (part-of-speech) a cada palabra de un texto, basándose en su función léxica y su contexto. Las etiquetas obtenidas son la base de otros análisis lingüísticos como el análisis sintáctico o semántico. Además, puede usarse para reconocimiento de nombres propios, desambiguación de sentidos de palabras o traducción automática.

   En la siguiente oración se muestran las etiquetas POS correspondientes a cada palabra: 'Alan (NNP) Turing (NNP) was (VBD) a (DT) brilliant (JJ) mathematician (NN)', donde NNP denota nombre propio singular, VBD denota un verbo en pasado, DT denota determinante y JJ denota un adjectivo.

   Los algoritmos de etiquetado POS de última generación son extremadamente precisos: obteniendo precisiones superiores al 97 % en 15 idiomas para el corpus UD y precisiones en torno al 97 % en varios corpus en inglés, sin importar el algoritmo (HMMs, CRFs, BERT). Estudios usando bi-LSTMs llegaron en 2016 también a un 96,5 % de acierto en 22 lenguajes.

9. **Explica cómo se representa el POS tagging como un Modelo oculto de Markov (HMM)**

   El POS tagging puede modelarse utilizando el Modelo oculto de Markov al asumir que las etiquetas POS son los estados no observados (ocultos) de un proceso de Markov X y las palabras de un texto son los valores observados de otro proceso de Markov Y que depende del proceso oculto X. El objetivo es encontrar la secuencia de estados ocultos (etiquetas) más probable que da lugar a la secuencia de palabras observadas utilizando el algoritmo Viterbi.

   Este modelo asume:

- La probabilidad de un estado particular depende sólo del estado anterior (Markov).
- La probabilidad de una observación de salida sólo depende del estado que produjo la observación y no de ningún otro estado ni de ninguna otra observación (output independence).

y se caracteriza por:

- Un conjunto $Q = \{t_1, ..., t_n\}$ de n estados, que son las etiquetas POS.
- Una matriz A que contiene las probabilidades de transición de las etiquetas $P(t_i|t_{i-1})$ que representan la probabilidad de que se produzca una etiqueta dada la etiqueta anterior.
- Una secuencia de T observaciones (palabras) $O = w_1 w_2 ... w_T$ del vocabulario V
- Una secuencia de observaciones verosímiles $B = b_i(w_T) = P(w_i|t_i)$, llamadas probabilidades de emisión, que expresan la probabilidad de que una palabra w_T haya sido generada por el estado (etiqueta) t_i.
- Un vector de probabilidades iniciales de los estados $\pi = (\pi_1, ..., \pi_n)$.

10. **Describe cuál es el objetivo del constituency parsing, bases de su funcionamiento y tipo de tareas para las que puede ser de utilidad.**

    De la idea de que grupos de palabras pueden comportarse como unidades únicas llamadas *constituyentes*, surgen las gramáticas libres de contexto que elaboran un inventario de los constituyentes de la lengua mediante palabras, símbolos y un conjunto de reglas que expresan las formas en que los símbolos del lenguaje pueden agruparse y ordenarse.

    Definidos los constituyentes y creadas las gramáticas libres de contexto, el constituency parsing tiene el objetivo de determinar si una oración puede derivarse de una gramática libre de contexto diviendo la oración en subfrases (constituyentes). Comienza con constituyentes de nivel inferior (tokens) y los agrupa en constituyentes mayores (frases).

    Constituency parsing es útil para varias tareas como el análisis semántico, la traducción automática y la extracción de información.

11. **Describe el problema de la ambigüedad en constituency parsing, qué consecuencias tiene y cómo se puede enfrentar el problema**

    El problema de la ambigüedad en constituency parsing surge debido a que algunas frases pueden tener múltiples derivaciones. Esto hace que haya muchos análisis gramaticales correctos pero muchos de ellos no tengan sentido semántico. Esto además tiene como consecuencia que el número de *parse trees* crece súper exponencialmente

    Para lidiar con el problema, solemos usar la programación dinámica. Esta nos ayuda a buscar en el espacio de posibles derivaciones de forma eficiente reutilizando estructuras compartidas localmente. Un ejemplo es el algoritmo CKY, que comprueba si una cadena está en un lenguaje sin utilizar todas posibles formas de parsear esa frase. Es un algoritmo bottom-up que se usa en gramáticas que estén en Forma Normal de Chomsky (CNF) (recordamos que toda gramática libre de contexto tiene una Gramática en CNF asociada que produce el mismo lenguaje), y que funciona del siguiente modo: primer construye todos los constituyentes más pequeños y luego trata de unirlos en constituyentes más grandes.

12. **Define, relaciona y compara brevemente constituency parsing y dependency parsing**

Constituency parsing es la tarea de analizar las oraciones dividiéndolas en subfrases, también conocidas como constituyentes. Esto permite determinar si la oración puede derivarse de una gramática libre de contexto y cómo. Comienza con constituyentes de nivel inferior (tokens) y los agrupa en constituyentes mayores (frases).

Una dependencia es una relación asimétrica sintáctica o semántica entre dos tokens léxicos. Así, dependency parsing es la tarea de extraer un árbol de dependencias de una frase, definiendo relaciones asimétricas sintácticas o semánticas entre palabras. En este caso, el árbol se construye añadiendo aristas entre tokens. Al igual que en constituency parsing, es útil para traducción automática y extracción de información, aunque también se puede utilizar para análisis de sentimientos y responder a preguntas.

Como diferencias, destacamos que las relaciones de dependencia no corresponden con relaciones de constituyencia, pues en esta última se agrupan los tokens mientras que en el dependency parsing se construye un grafo añadiendo vértices entre los tokens. Además, es más rápido y más independiente del lenguaje que otras tareas de parsing.

13. **Define, relaciona y compara brevemente aproximaciones basadas en transiciones y aproximaciones basadas en grafos de dependency parsing**

    En la tarea de **dependency parsing**, se usan principalmente dos algoritmos, aunque ambos tengan limitaciones computacionales: basadas en transiciones y basadas en grafos. Ambos enfoques utilizan datos etiquetados o no etiquetados para entrenar un modelo que realice un análisis sintáctico.

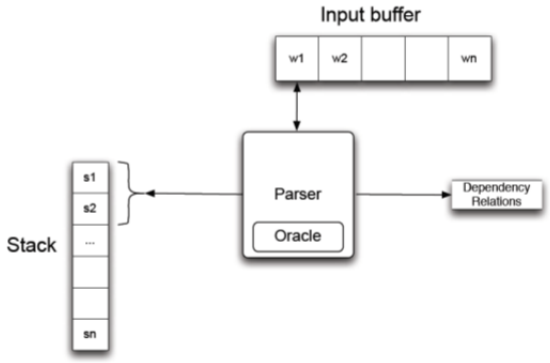
    Los enfoques basados en transiciones son métodos locales que buscan relaciones entre pares de palabras y utilizan operaciones como desplazamiento o reducción. Producen árboles proyectivos, lo que hace que se produzcan errores en las oraciones que tengan estructuras no proyectivas. Realizan búsquedas codiciosas y encuentran óptimos locales. El coste de estos métodos para oraciones proyectivos es $O(n)$ y $O(n^2)$ para no proyectivas.

    Por su parte, los enfoques basados en grafos son más flexibles y solventan la limitación anterior para oraciones no proyectivas. Estos métodos realizan un análisis global construyendo un grafo completo con aristas dirigidas y ponderadas, lo que permite encontrar el árbol de expansión con la mayor puntuación (suma de todas las aristas ponderadas). Realizan búsquedas exhaustivas y encuentran óptimos globales. El coste de estos métodos para oraciones proyectivas aumenta respecto al enfoque basado en transiciones, siendo $O(n^3)$ y $O(n^2)$ para no proyectivas.

14. **Describe brevemente la arquitectura y funcionamiento generales de un dependency parser basado en transiciones.**

    Un dependency parser basado en transiciones es un método local que extrae un árbol de relaciones asimétricas, tanto sintácticas como semánticas, entre pares de palabras de una oración utilizando tres tipos de acciones principales: shift, left y right. Shift decide si cambiar el foco a la siguiente pareja de palabras, y left/right decide si la palabra de la izquierda/derecha depende de la palabra de la derecha/izquierda. Este tipo de enfoques están basados en el análisis sintáctico *shift-reduce*, desarrollado originalmente para analizar lenguajes de programación. Su configuración requiere utilizar:

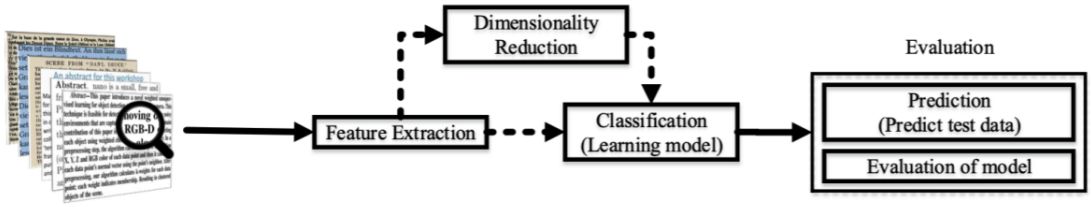
    - pila: inicializada con el nodo ROOT
    - entrada: inicializada con palabras de una oración
    - salida: conjunto de relaciones de dependencia



Las relaciones de dependencia son obtenidas utilizando un analizador sintáctico que examina los dos primeros elementos de la pila y selecciona una acción basada en la consulta de un oráculo que examina la configuración actual.

15. **Describe la cadena de procesamiento (pipeline) de la clasificación de texto, explicando cada una de sus fases.**

    Un pipeline de clasificación de texto se puede esquematizar como se muestra:



    La primera tarea consiste en **extraer características** convertiendo texto no estructurado en un espacio de características estructurado utilizando modelos matemáticos como TF-IDF o Word2Vec.

    A continuación, de manera opcional puede realizarse una **reducción de la dimensionalidad** del vector de características para reducir el tiempo y la complejidad del modelo de clasificación de texto. Las técnicas más utilizadas son PCA, LDA y t-SNE, entre otras.

    A continuación se elige un **modelo de aprendizaje** para clasificar el texto de acuerdo a la tarea que queramos llevar a cabo. En general, se utilizan Naïve Bayes (NBC) y SVM como clasificadores base o baseline. Boosting y Bagging se utilizan en estrategias de aprendizaje de consultas y análisis de textos. Árboles de Decisión y Random Forest se utilizan para categorizar documentos y enfoques de deep learning se utilizan para modelar relaciones complejas y no lineales de los datos.

    Definido el modelo de aprendizaje se entrena sobre un conjunto de datos de prueba y se obtienen las predicciones para un conjunto de datos de prueba. El último paso consiste en **evaluar** los resultados comparándolos con las etiquetas reales de los datos de prueba. Se utilizan métricas para comprender el rendimiento de un modelo de clasificación de textos, destacando $F\beta$ score y área bajo la curva ROC (AUC), entre otras.

16. **Describe en qué consiste el Análisis Discriminante Lineal (LDA) y sus objetivos.**

    El Análisis Discriminante Lineal (LDA) es una técnica de reducción de dimensión que, aplicada a procesamiento de lenguaje natural, permite para modelar temas en documentos y representarlos en un espacio de temas de menor dimensión.

LDA considera cada documento como una mixtura de temas del corpus y cada tema como una distribución de palabras clave asociadas a cada tema. De esta manera, las palabras clave de un documento permiten identificar de qué tema trata el documento.

Por tanto, el objetivo es conseguir una buena composición de la distribución *tema-palabras clave* para un número de temas reordenando la distribución de los temas dentro de los documentos y la distribución de las palabras clave dentro de los temas.

Los factores clave de LDA son:

- La calidad del tratamiento del texto.
- La variedad de temas de los que habla el texto.
- La elección del algoritmo de modelización de temas.
- El número de temas introducidos en el algoritmo.
- Los parámetros de ajuste del algoritmo.

17. **Describe en qué consiste el método de Naïve Bayes para la clasificación de textos, las presunciones y simplificaciones en las que se basa, y sus posibles optimizaciones para la minería de opiniones.**

    Naive Bayes (NB) es un clasificador probabilístico que encuentra la clase $\hat{c} \in C$ asociada a un documento d que maximiza la probabilidad a posteriori:

    $$\hat{c} = \arg\max P(c|d)$$

    Las presunciones/simplificaciones en las que se basa son las siguientes:

    - Un documento d se representa como un conjunto de características $f_1, ..., f_n$ con probabilidades independientes tal que, $P(f_1, \ldots, f_n|c) = P(f_1|c) \ldots P(f_n|c)$.
    - Las palabras se indexan por posición.
    - Se utiliza el logaritmo por temas de eficiencia.

    Para mejorar el rendimiento en la minería de opiniones, se utiliza NB binario. En este modelo no se tiene en cuenta la frecuencia de una palabra en un documento, sino el mero hecho de aparecer. Si lo hace, contará como 1 y si no, como 0. Además, cada vez que aparezca un token de negación, le añadiremos la palabra NOT a todas las palabras entre ese token y el siguiente signo de puntuación. Es decir, *didn't like this movie, but* pasaría a ser *didn't NOT_like NOT_this NOT_movie, but*. Esto nos ayudará a percibir con mayor facilidad un sentimiento negativo.

18. **Describe qué se entiende por análisis de emociones y los distintos modelos que existen para implementarlo.**

    El análisis de emociones es una tarea que está dentro del análisis de opinión o *sentiment analysis* y consiste en la detección automáticas de las emociones en texto con el objetivo de mejorar tareas de procesamiento de lenguaje: mejora de sistemas de diálogo, detección de depresión, satisfacción de clientes en hoteles, etc.

    Hay 3 modelos principales para analizar emociones.

    - El modelo de Ekman categoriza las emociones en 6 tipos básicos que pueden generar otros tipos: sorpresa, desagrado, felicidad, ira, miedo y tristeza.
    - Otro modelo fue propuesto por Plutchik y se define como rueda de 8 emociones emparejadas por pares opuestos: alegría-tristeza, ira-miedo, confianza-desagrado y anticipación-sorpresa, junto con las emociones derivadas de ellas.

- El último modelo se basa en la idea de que las palabras varían en tres dimensiones de significado afectivo:
  - Valencia: carácter agradable de un estímulo.
  - Excitación: intensidad de la emoción provocada por un estímulo.
  - Dominación: grado de control ejercido por un estímulo.

  Este modelo asume que la dimensión de valencia permite medir el sentimiento de una palabra, de acuerdo a lo agradable o desagradable que sea.

19. **Explica en qué consiste la extracción de información en PLN, mencionando sus principales tareas.**

    La extracción de información (EI) es la tarea de extraer información relevante de los documentos (entidades, relaciones y eventos) con el objetivo de construir de manera automática una base de conocimiento estructurada mediante el procesamiento de texto en lenguaje natural. La EI ha surgido de la investigación sobre sistemas basados en reglas de la Lingüística Computacional y utiliza análisis lingüísticos sobre el texto original, que puede ser estructurado o semiestructurado y puede pertenecer a un solo documento o a varios. Se evalúa de acuerdo a lo correcta que sea la base de conocimiento resultante.

    No hay que confundirla con Information Retrieval (IR), que trata de extraer **documentos relevantes** de conjunto de documentos.

    Las principales tareas que envuelve son:

    - Reconocimiento de entidades con nombres.
    - Reconocimiento de expresiones temporales utilizando enfoques basados en secuencias o normalizaciones temporales.
    - Extracción y vinculación de relaciones entre entidades.
    - Detección de eventos (extracción, clasificación y ordenación temporal).
    - Relleno de plantillas con información de los eventos detectados.

20. **Da ejemplos de bases de conocimiento que se usan en PLN, explicando brevemente cómo se pueden usar en tareas de extracción de información.**

    El objetivo final de la extracción de información (IE) es la de crear un sistema de construcción automática una base de conocimiento estructurado mediante herramientas de Procesamiento del Lenguaje Natural, donde la entrada de texto puede ser:

    - Información no estructurada o semiestructurada.
    - Un documento o múltiples documentos.

    Las tareas IE sobre las que se aplican las bases de conocimiento podemos encontrar las siguientes:

    - Named Entity Recognition (NER): identifica si los *spans* (secuencia de tokens) de los textos son entidades identificadas como ORGANIZATION, LOCATION o DATE, por ejemplo.
    - Named Entity Linking (NEL): relaciona específicamente los spans identificados en una sola base de conocimiento. (Ver más en Wikipedia). También se denomina *Wikification*.

    Dentro de las bases de conocimiento que se utilizan para estas causas se encuentran:

- **DBpedia**: conocida como la ontología de Wikipedia por excelencia. Tiene 768 clases, 3000 propiedades y 4.8 millones de instancias. Está compuesta por tuplas de la forma `<subject, property, object>`, obtenidas por contenido semiestructurado procedente de la Wikipedia (cuadros de información, categorías, listas o tablas). Está basado en formato XML (RDFs) y lenguaje de queries SPARQL. En concreto, las propiedades pueden ser una única URI seguida por un alias. Se construye, además, sobre un modelo basado en grafos dirigidos que expresan relaciones semánticas. Podemos ver un ejemplo de esto en la Figura 1

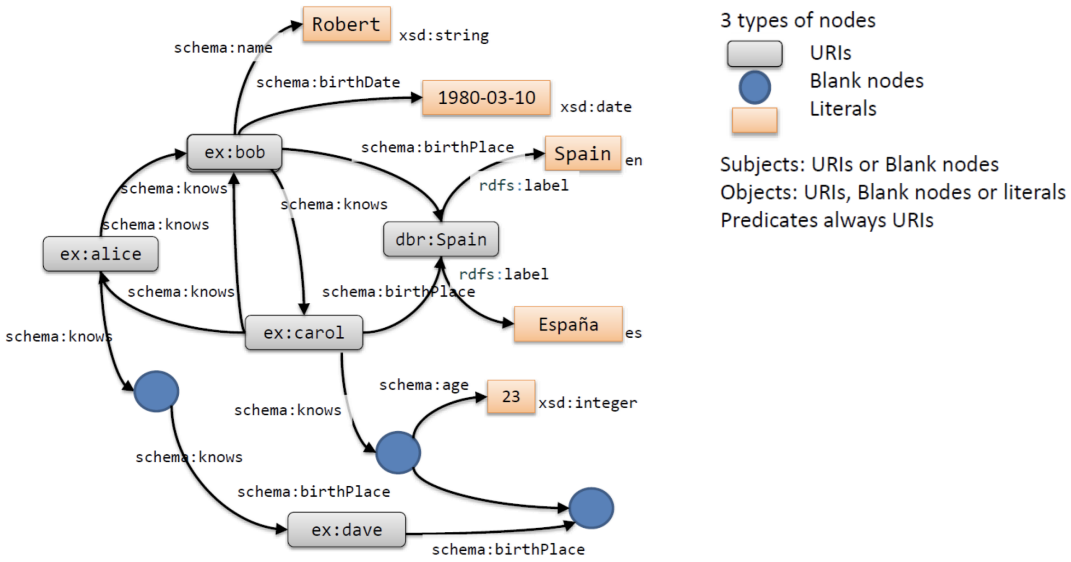


Figura 1: Ejemplo de un modelo de datos RDF basado en grafos dirigidos.

- **WordNet**: base de datos léxica para el inglés con ciertas relaciones entre entidades.
- **YAGO**: base de conocimiento open-source con más de 10 millones de entidades extraídas de Wikipedia, WordNet y GeoNames.
- **Freebase**: la cual ahora es Google Knowledge Graph.
- **Linked Open Data (LOD)**: web completa de objetos interrelacionados en formato RDF.
- **Desambiguación de Wikipedia**: se puede utilizar como recurso para la selección de términos que tengan más de un significado.

Idea: se puede utilizar, por ejemplo en áreas en donde se requiera clasificar usuarios. Cuando necesitas asignar temáticas al contenido que publica un usuario se pueden utilizar estas bases de conocimiento para relacionar conceptos.