TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Diseño e implementación de un analizador de dependencias para procesamiento de lenguaje natural en Español

Mediante Máquinas de Soporte Vectoriales

Autor
Alejandro Alcalde Barros

Directores
Salvador García

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

20 de noviembre de 2016

Alejandro Alcalde Barros: *Diseño e implementación de un analizador de dependencias para procesamiento de lenguaje natural en Español,* Mediante Máquinas de Soporte Vectoriales, Grado en Ingeniería Informática, © 20 de noviembre de 2016

TUTOR:
Salvador García

LOCALIZACIÓN:
Granada

ÚLTIMA MODIFICACIÓN:
20 de noviembre de 2016

Ohana means family.
Family means nobody gets left behind, or forgotten.

— Lilo & Stitch

Dedicated to the loving memory of Rudolf Miede.
1939 – 2005

RESUMEN

En este trabajo se implementa un método para analizar dependencias palabra a palabra con una estrategia de abajo a arriba (*Bottom-Up*) usando Máquinas de Soporte Vectoriales (*SVMs*). En concreto, este trabajo se ha centrado en analizar las dependencias entre palabras en Castellano. Aunque la precisión de los resultados no está cerca del estado del arte, es necesario tener en cuenta que este parseador no usa información sobre la estructura de las frases.

ETIQUETAS: PNL, SVM, Parseo de dependencias.

ABSTRACT

In this project, a method for analyzing word to word dependencies is implemented using a bottom-up strategy with the help of Support Vector machines. In particular, this work has focused in analyzing dependencies between words for spanish language. Even though accuracy is far from the state of the art, it is worth noting this parser is not using information about the sentences structure.

TAGS: NLP, SVM, Dependency parsing.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [3]

ACKNOWLEDGMENTS

Put your acknowledgments here.

Many thanks to everybody who already sent me a postcard!

Regarding the typography and other help, many thanks go to Marco Kuhlmann, Philipp Lehman, Lothar Schlesier, Jim Young, Lorenzo Pantieri and Enrico Gregorio[1], Jörg Sommer, Joachim Köstler, Daniel Gottschlag, Denis Aydin, Paride Legovini, Steffen Prochnow, Nicolas Repp, Hinrich Harms, Roland Winkler, Jörg Weber, Henri Menke, Claus Lahiri, Clemens Niederberger, Stefano Bragaglia, Jörn Hees, and the whole LaTeX-community for support, ideas and some great software.

Regarding LyX: The LyX port was intially done by *Nicholas Mariette* in March 2009 and continued by *Ivo Pletikosić* in 2011. Thank you very much for your work and for the contributions to the original style.

1 Members of GuIT (Gruppo Italiano Utilizzatori di TeX e LaTeX)

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

LISTINGS

ACRONYMS

NLP Natural Language Processing

PNL Procesamiento del Lenguaje Natural

IA Inteligencia Artificial

ASR Automatic Speech Recognition

RVA Reconocimiento de Voz Automático

POS Part-Of-Speech

AA Aprendizaje Automático

NER Named Entities

DP Dependency Parsing

HRL High-Resource Language

LRL Low-Resource Language

SDS Spoken Dialogue System

DM Dialogue Management

TTS Text-To-Speech

API Application Programming Interface

Parte I

PUESTA EN ESCENA

En este primer apartado expondremos el conocimiento previo necesario para que el lector se situe en el contexto del problema tratado.

1

MOTIVACIÓN E INTRODUCCIÓN

TODO LIST

Pipeline Genérico, detallar cada parte, con ejemplos y justificación

Pipeline Específico, con paquete real (CoreNLP), explicar de forma breve cada parte, inglés)

Desribir de forma más detallada el Dep Parsing, mencionar state of the art (3/4 papers), entre ellos el implementado

Sección con algoritmo implementado, reiterando sección anterior pero con lujo de detalles (Teóricos y código)

Motivación: Falta de software español, justificar decisión de afrontar problema.

Añadir sección "*El resto del paper está organizado...*"

1.1 ¿QUÉ ES EL PROCESAMIENTO DEL LENGUAJE NATURAL?

El lenguaje natural se refiere a cualquier lenguaje hablado por un humano, (por ejemplo,Inglés, Castellano o Chino). El *Natural Language Processing* (NLP) es un campo de la ciencia de la computación e ingeniería desarrollado a partir del estudio del lenguaje y la computación lingüistica dentro del campo de la Inteligencia Artificial (IA).

Procesamiento del Lenguaje Natural (PNL)

Los objetivos del NLP son diseñar y construir aplicaciones que faciliten la interacción humana con la máquinas y otros dispositivos mediante el uso del lenguaje natural. Dentro del amplio campo del NLP podemos distinguir las siguientes áreas principales:

Procesamiento del Lenguaje Natural (PNL)

El lenguaje natural se refiere a cualquier lenguaje hablado por un humano, (por ejemplo,Inglés, Castellano o Chino). El NLP es un campo de la ciencia de la computación e ingeniería desarrollado a partir del estudio del lenguaje y la computación lingüistica dentro del campo de la IA. Los objetivos del NLP son diseñar y construir aplicaciones que faciliten la interacción humana con la máquinas y otros dispositivos mediante el uso del lenguaje natural. Dentro del amplio campo del NLP podemos distinguir las siguientes áreas principales:

RESÚMENES este área incluye aplicaciones que puedan, basándose en una colección de documentos, dar como salida un resumen coherente del contenido de los mismos. Otra de las posibles aplicaciones sería generar presentaciones a partir de dichos documentos. En los últimos años, la información disponible en la red ha aumentado considerablemente. Un claro ejemplo es la literatura científica, o incluso repositorios de información más genérica como *Wikipedia*. Toda esta información escrita en lenguaje natural puede aprovecharse para entrenar modelos que sean capaces de generar hipótesis por sí mismos, generar resúmenes o extraer hechos. Un ejemplo claro puede ser la extracción de hechos básicos que relacionen dos entidades (*"Luís es padre de Cristina"*.

TRADUCCIÓN AUTOMÁTICA: Esta fue la principal área de investigación en el campo del NLP. Como claro ejemplo tenemos el traductor de Google, mejorando día a día. Sin embargo, un traductor realmente útil sería aquel que consiga traducir en tiempo real una frase que le dictemos mientras decidimos qué línea de autobús debemos coger para llegar a tiempo a una conferencia en Zurich. La traducción entre lenguajes es quizá una de las formas más transcendentales en las que las máquinas podrían ayudar en comunicaciones entre humanos. Además, la capacidad de las máquinas para traducir entre idiomas humanos se considera aún como un gran test a la IA, ya que una traducción correcta no consiste en el mero hecho de generar frases en un idioma humano, también requiere del conocimiento humano y del contexto, pese a las ambigüedades de cada idioma. Por ejemplo, la traducción literal *"bordel"* en Francés significa Burdel; pero si alguien dice *"Mi cuarto es un burdel"*, el traductor debería tener el conocimiento suficiente para inferir que la persona se está refiriendo a que su habitación es un desorden.

Debería buscar algún ejemplo más claro?

La traducción automática fue una de las primeras aplicaciones no numéricas de la computación y comenzó a estudiarse de forma intensiva en la década de los 50. Sin embargo, no fue hasta la década de

los 90 cuando se produjo una tansformación en este área. IBM se hizo con una gran cantidad de frases en Inglés y Francés que eran traducciones las unas de las otras, lo cual permitió recopilar estadísticas de traducciones de palabras y secuencias de palabras, concediento así el desarrollo de modelos probabilísticos para la traducción automática. Hasta ese momento, todo el análisis gramático se hacía manualmente.

Conocido como texto paralelo

A la llegada del nuevo milenio, se produjo una explosión de texto disponible en la red, así como grandes cantidades de *texto paralelo*. Se dieron invención a nuevos sistemas para la traducción automática basados en modelos estadísticos basados en frases en lugar de palabras. En lugar de traducir palabra por palabra, ahora se tenían en cuenta pequeños grupos de palabras que a menudo poseen una traducción característica.

En los últimos años, y mediante el uso de *deep learning* se están desarrollando modelos de secuencias basados en este tipo de aprendizaje bastante prometedores. La idea principal del *deep learning* reside en entrenar un modelo con varios niveles de representación para optimizar el objetivo deseado, una traducción de calidad, en este caso. Mediante estos niveles el modelo puede aprender representaciones intermedias útiles para la tarea que le ocupa. Este método de aprendizaje se ha explotado sobre todo en redes neuronales. Un ejemplo claro de *deep learning* usando redes neuronales es el reconocimiento de dígitos, cada capa interna de la red neuronal intenta extraer características representativas de cada dígito a distintas escalas. Podemos ver una demostración de este comportamiento en Pound y Riley [8]

RECONOCIMIENTO DE VOZ: Una de las tareas más difíciles en NLP. Aún así, se han conseguido grandes avances en la construcción de modelos que pueden usarse en el teléfono móvil o en el ordenador. Estos modelos son capaces de reconocer expresiones del lenguaje hablado como preguntas y comandos. Desafortunadamente, los sistemas *Automatic Speech Recognition* (ASR) funcionan bajo dominios muy acotados y no permiten al interlocutor desviarse de la entrada que espera el sistema, por ejemplo,*"Por favor, diga ahora la opción a elegir: 1 Para..., 2 para..."*

Reconocimiento de Voz Automático (RVA)

SDS: Los *Spoken Dialogue Systems (SDSs)*. El diálogo ha sido un tema popular para el NLP desde los 80. En estos sistemas se pretende remplazar a los usuales buscadores en los que introducimos un texto para obtener algún tipo de respuesta a una pregunta. Por ejemplo, si quisieramos saber a qué hora abre un centro comercial, bastaría con hablarle al sistema en lenguaje natural – nuestro lenguaje natural, ya sea Inglés, Alemán o Castellano y el sistema nos daría respuesta a nuestra pregunta. Aunque ya existen este tipo de sistemas (por ejemplo,*Siri de Apple, Cortana de Microsoft, Google Now...*) están aún en

SDS: Sistemas de Diálogo Hablados

una situación muy precaria, ya que ninguno entiende por completo el lenguaje natural, solo un subconjunto de frases clave.

La creación de SDSs, ya sea entre humanos o entre humanos y agentes artificiales requiere de herramientas como:

- ASR, para identificar qué dice el humano.
- *Dialogue Management* (DM), para determinar qué quiere el humano.

DM: Gestión del diálogo

- Acciones para obtener la información o realizar la actividad solicitada.
- Síntesis *Text-To-Speech* (TTS), para comunicar dicha información al humano de forma hablada.

Leer un texto por una máquina

En la actualidad, Geoffrey Hinton [1] desarrollaron un SDS haciendo uso de *deep learning* para mapear señales sonoras a secuencias de palabras y sonidos del idioma humano, logrando avances importantes en la precisión del reconocimiento del habla.

CLASIFICACIÓN DE DOCUMENTOS: Una de las áreas más exitosas del NLP, cuyo objetivo es identificar a qué categoría debería pertenecer un documento. Ha demostrado tener un amplio abanico de aplicaciones, por ejemplo,filtrado de *spam*, clasificación de artículos de noticias, valoraciones de películas... Parte de su éxito e impacto se debe a la facilidad relativa que conlleva entrenar los modelos de aprendizaje para hacer dichas clasificaciones.

ANÁLISIS DE SENTIMIENTOS: Gran parte del trabajo en NLP se ha centrado en el análisis de sentimientos (identificación de orientaciones positivas o negativas en textos) e identificación de creencias positivas, negativas o neutrales en frases basándose en información léxica y sintáctica. Tanto las creencias como los sentimientos constitiyen actitudes hacia eventos y proposiciones, aunque en concreto, los sentimientos pueden también referirse a actitudes hacia objetos tales como personas, organizaciones y conceptos abstractos. La detección de sentimientos y emociones en texto requiere de información léxica y a nivel de la propia sentencia. Por lo general, el sentimiento puede detectarse a través del uso de palabras expresando orientaciones positivas o negativas, por ejemplo,*triste, preocupado, difícil* son todas palabras con una connotación negativa, mientras que *cómodo, importante, interesante* denotan un sentimiento positivo. Las aproxmiaciones más sofisticadas para el anlálisis de sentimientos intentan buscar tanto la fuente como el objeto del sentimiento, por ejemplo,quién está expresando un sentimiento positivo sobre alguna persona, objeto, actividad o concepto.

La comunidad del reconocimiento de voz está igualmente implicada en el estudio de actitudes positivas y negativas, centrándose en la

identifiación de emociones haciendo uso de información acústica y prosódica. Otras investigaciones se han centrado en identificar emociones particulares, específicamente las séis emociones básicas según Ekman – ira, aversión, temor, dicha, tristeza y asombro – las cuales pueden ser reacciones a eventos, proposiciones u objetos. Por contra, la generación de emociones ha demostrado ser un reto mucho mayor para la síntesis TTS.

Un acento prosódico

¿Definir palabra?

La clasificación de sentimientos es algo ampliamente usado para identificar opiniones – puntos de vista positivos o negativos hacia personas, instituciones o ideas – en muchos idiomas y géneros. Una de las aplicaciones más prácticas, y de las que más abundan consiste en identificar críticas sobre películas o productos [7, 9].

La minería de datos en redes sociales con el fin de realizar análisis de sentimientos se ha convertido en un tema popular con el objetivo de evaluar el *estado de ánimo* del público – de twitter, por ejemplo. –

El NLP emplea técnicas computacionales con el propósito de aprender, entender y producir lenguaje humano. Las aproximaciones de hace unos años en el campo de la investigación del lenguaje se centraban en automatizar el análisis de las estructuras lingüísticas y desarrollar tecnologías como las mencionadas anteriormente. Los investigadores de hoy en día se centran en usar dichas herramientas en aplicaciones para el mundo real, creando sistemas de diálogo hablados y motores de traducción *Speech-to-Speech*, es decir, dados dos interlocutores, interpretar y traducir sus frases. Otro de los focos en los que se centran las investigaciones actuales son la minería en redes sociales en busca de información sobre salud, finanzas e identificar los sentimietos y emociones sobre determinados productos.

1.2 HISTORIA DEL PROCESAMIENTO DEL LENGUAJE NATURAL

A continuación, citamos algunos de los avances en este campo durante los últimos años según Hirschberg y Manning [2].

Durante las primeras épocas de esta ciencia, se intentaron escribir vocabularios y reglas del lenguaje humano para que el ordenador las entendiera. Sin embargo, debido a la naturaleza ambigua, variable e interpretación dependiente del contexto de nuestro lenguaje resultó una ardua tarea. Por ejemplo, una estrella puede ser un ente astronómico o una persona, y puede ser un nombre o un verbo.

En la década de los 90, los investigadores transformaron el mundo del NLP desarrollando modelos sobre grandes cantidades de datos sobre lenguajes. Estas bases de datos se conocen como *corpus*. El uso de estos conjuntos de datos fueron uno de los primeros éxitos notables del uso del *big data*, mucho antes de que el Aprendizaje Automático (AA) acuñara este término.

Esta aproximación estadística al NLP descubrió que el uso de métodos simples usando palabras, secuencias del *Part-Of-Speech* (POS)

POS: Categorías morfosintácticas en castellano

(si una palabra es un nombre, verbo o preposición), o plantillas simples pueden obtener buenos resultados cuando son entrenados sobre un gran conjunto de datos. A día de hoy, muchos sistemas de clasificación de texto y sentimientos se basan únicamente en los distintos conjuntos de palabras o *"bag of words"* que contienen los documentos, sin prestar atención a su estructura o significado. El estado del arte de hoy día usa aproximaciones con AA y un rico conocimiento de la estructura lingüística. Un ejemplo de estos sistemas es *Stanford* CoreNLP [6]. CoreNLP proporciona un *pipeline* estándar para el procesamiento del NLP incluyendo:

POS TAGGING: Etiquetado morfosintáctico. Módulo encargado de leer texto en algún lenguaje y asignar la categoría morfosintáctica a cada palabra, por ejemplo,nombre, verbo, adjetivo... aunque por lo general se suelen usar etiquetas más detalladas como "*nombre-plural*".

NER: *Named Entities* (NER), etiqueta palabras en un texto correspondientes a *nombres de cosas*, como personas, nombres de compañías, nombres de proteínas o genes etc. En concreto, CoreNLP distingue de forma muy precisa tres tipos de clases, personas, organizaciones y localizaciones.

PARSEO GRAMATICAL: Resuelve la estructura gramatical de frases, por ejemplo,qué grupos de palabras van juntos formando frases y qué palabras son sujeto u objeto de un verbo. Como se ha comentado, en aproximaciones anteriores se usaban parseadores probabilísticos usando conocimiento del lenguaje a partir de sentencias analizadas sintácticamente a mano. Para así producir el análisis más probable de sentencias nuevas. Actualmente se se usan parseadores estadísticos, los cuales aún comenten fallos, pero funcionan bien a rasgos generales.

DP: *Dependency Parsing* (DP) o parseo de dependencias. Analiza la estructura gramatical de una frase, estableciendo relaciones entre palabras principales y palabras que modifican dichas palabras principales. La Figura 1 muestra un ejemplo. La flecha dirigida de la palabra *moving* a la palabra *faster* indica que *faster* modifica a *moving*. La flecha está etiquetada con una palabra, en este caso *advmod*, indicando la naturaleza de esta dependencia. La Figura 2 muestra ejemplos de los distintos módulos del *pipeline* de CoreNLP

1.3 LIMITACIONES

Aunque se han producido avances, una de las principales limitaciones del NLP hoy día es el hecho de que la mayoría de recursos y sistemas solo están disponibles para los denominados *High-Resource*

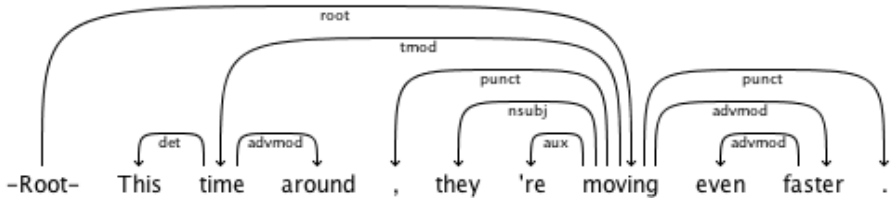

Figura 1: Ejemplo de parseo de dependencias

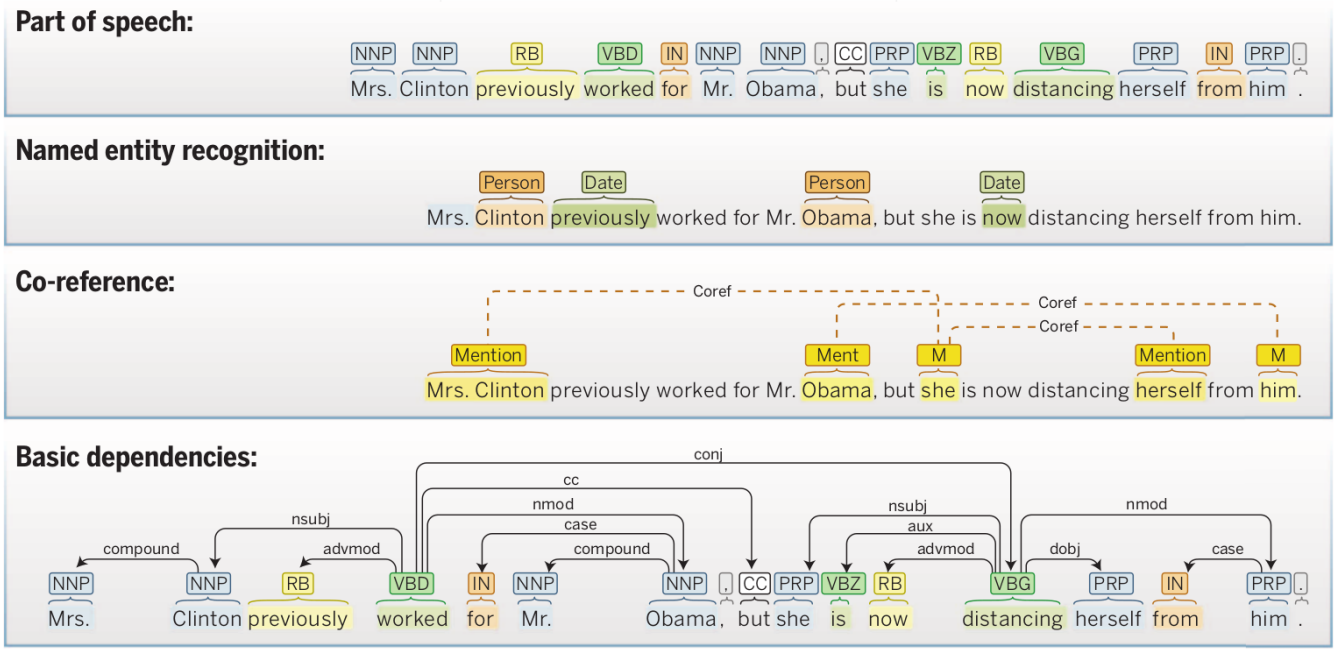

Figura 2: Many language technology tools start by doing linguistic structure analysis. Here we show output from Stanford CoreNLP. As shown from top to bottom, this tool determines the parts of speech of each word, tags various words or phrases as semantic named entities of various sorts, determines which entity mentions co-refer to the same person or organization, and then works out the syntactic structure of each sentence, using a dependency grammar analysis.

Languages (HRLs), estos lenguajes son el Inglés, Francés, Español, Alemán y Chino. Por contra, hay una gran cantidad de *Low-Resource Languages (LRLs)* – como Bengalí, Indonesio, Punjabí, Cebuano y Swahili – hablados y escritos por millones de personas que no disponen de este tipo de sistemas. Uno de los mayores retos para la comunidad del lenguaje es desarrollar recursos y herramientas para cientos o miles de lenguajes, no solo para unos pocos.

HRL: Idiomas de altos recursos

LRL: Idiomas de bajos recursos

Aún existiendo bastante *software* trabajando con NLP, y para idiomas HRL, suelen obtenerse mejores resultados para un idioma en concreto, el Inglés. Es por ello que este trabajo se ha centrado en desarrollar una fase del *pipeline* que se encuentra en todos los sistemas que realizan análisis de sentimientos, y en general NLP para el idioma Español. Como ejemplo podemos citar el famoso CoreNLP [5].

Me dijiste "*justificar el por qué hemos decidido afrontar este problema*", pero no se me ocurre qué poner

Cuadro 1: *Pipeline* de CoreNLP y disponibilidad por lenguaje

ANNOTATOR	AR	ZH	EN	FR	DE	ES
Tokenize / Segment	✓	✓	✓	✓		✓
Sentence Split	✓	✓	✓	✓	✓	✓
Part of Speech	✓	✓	✓	✓	✓	✓
Lemma			✓			
Named Entities		✓	✓		✓	✓
Constituency Parsing	✓	✓	✓	✓	✓	✓
Dependency Parsing		✓	✓	✓	✓	
Sentiment Analysis			✓			
Mention Detection		✓	✓			
Coreference		✓	✓			
Open IE			✓			

En la Tabla 1 se lista todo el *pipeline* de CoreNLP junto con el soporte para cada lenguaje. Como se aprecia, el *pipeline* esta completo únicamente para el Inglés. El objetivo de este trabajo ha consistido en implementar un parseo de dependencias para el Español.

Con la introducción del *pipeline* de CoreNLP, se pasa ahora a describir el proceso que todo sistema para NLP debe seguir. Comenzaremos mencionando un proceso genérico, para después profundizar en el *pipeline* de un *software* especícifo, CoreNLP en nuestro caso.

1.4 EL PIPELINE GENÉRICO

En esta sección se comentará el proceso habitual que suele seguirse como *pipeline* en los problemas de NLP. Para ello se describirán los distintos niveles de análisis en los que opera dicho *pipeline*, así como las diferentes aproximaciones que se usan y los problemas más comunes a los que se enfrenta todo sistema que realice análisis de sentimientos.

Me sigo refiriendo a NLP, o menciono *análisis de sentimientos/opinion Mining*?

Liu [4] define una opinión como una quíntupla conteniendo el objetivo de la opinión (o *entidad*), el atributo del objetivo al que se dirige la opinión, el sentimiento (o polaridad) de la opinión, pudiendo ser este positivo, negativo o neutral, el poseedor de dicha opinión y la fecha en la que se produjo. Formalmente se podría definir como la tupla:

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$$

donde e_i corresponde con el objetivo de la opinión i-ésima, a_{ij} es el j-ésimo atributo de e_i, h_k el k-ésimo poseedor de la opinión, t_l codifica el tiempo en el que se emitió la opinión y por último, s_{ijkl} es

la polaridad de la opinión hacia el atributo a_{ij} para la entidad e_i por el poseedor de la opinión h_k en el momento t_l.

El principal objetivo del análisis de sentimientos consiste en encontrar todas las tuplas $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ en un documento o colección de documentos.

1.4.1 *Pasos previos*

El procesamiento más usual para realizar tareas de análisis de sentimietos se puede dividir en una serie de pasos definidos. Dichos pasos corresponden a la adquisición del *corpus* o datos, preprocesamiento del texto, el proceso principal del análisis de sentimientos, agregación y resumen de los resultados y por último, visualización. En los próximos apartados se mencionarán los tres primeros.

1.4.1.1 *Adquisición de Datos*

En este paso se debe obtener el *corpus* para el cual se desea realizar el análisis de sentimientos. Actualmente existen dos aproximaciones para realizar esta tarea. Una de ellas consiste en hacer uso de la *Application Programming Interface* (API) de alguna página web de la que se desee extraer el *corpus*. Una de las APIs más populares para este propósito es la de Twitter. La segunda aproximación hace uso de *Web Crawlers* para extraer datos de las webs deseadas.

Rastreadores de Webs

Ambas aproximaciones presentan sus ventajas y desventajas, y por tanto existen *trade-off* en función de cual se decida usar. Veamos algunos de ellos.

Conozco bien el concepto de esta palabra, pero no encuentro la adecuada traducción. ¿Sacrificios?. Podría poner, *En función de qué aproximación se decida usar, sacrificaremos unas ventajas frente a otras*

Mediante el uso de una API la implementación es sencilla, los datos obtenidos están ordenados y poseen una estructura poco sujeta al cambio, sin embargo, en función del proveedor de la API se presentan ciertas limitaciones. Siguiendo con Twitter, su API limita a 180 consultas cada 15 minutos el número de peticiones que se pueden realizar. Además, su API para *streaming* presenta otras limitaciones. En lugar de imponer límites a la cantidad de peticiones, restringe el número de clientes que se pueden conectar desde la misma dirección IP al mismo tiempo, así como la velocidad a la que cada uno puede leer los datos. Pese a las limitaciones anteriores, la más importante quizás sea que esta aproximación depende de la existencia de una API por parte del sitio web.

Por otro lado, la aproximación basada en rastreadores webs son bastante más complejas de implementar, la razón principal se debe a que los datos obtenidos, por norma general tendrán ruido y no estarán estructurados. Como beneficio, esta aproximación tiene la capacidad no imponernos prácticamente ninguna restricción. Si bien es cierto que se deben respetar ciertas normas y protocolos, como las

indicaciones del fichero ROBOTS.TXT[1] de cada sitio web, no realizar múltiples peticiones al mismo servidor y espaciar las mismas para no someter al servidor a demasiada carga.

1.4.1.2 *Preprocesamiento del texto*

El segundo paso en el *pipeline* del análisis de sentimientos es el preprocesamiento del texto adquirido. En este paso se realizan varias tareas habituales para el NLP correspondientes al análisis léxico. Algunas de estas tareas son:

TOKENIZACIÓN: Encargada de separar las cadenas de texto del documento completo en una lista de palabras. Es muy sencilla de realizar para idiomas delimitados por espacios como el Inglés, Español o Francés, pero se torna considerablemente más compleja para idiomas donde las palabras no son delimitadas por espacios, como el Japonés, Chino y Thai.

STEMMING: Proceso heurístico encargado de eliminar los afijos de la palabra para dejarlos en su forma canónica (invariante, o raíz). Por ejemplo, *persona, personificar y personificación* pasan a ser *persona* una vez acabado este proceso.

traducción literal de *non-inflected dictionary form*

LEMATIZACIÓN: Proceso algorítmico para convertir una palabra a su forma de diccionario no-inflexible. Esta fase es análoga a la anterior (*stemming*) pero se realiza a través de una serie de pasos más rigurosos que incorporan un análisis morfológico de cada palabra.

ELIMINAR STOPWORDS: Actividad encargada de borrar las palabras usadas para estructurar el lenguaje pero que no contribuyen de modo alguno a su contenido. Algunos ejemplos de estas palabras pueden ser *de, la, que, el, en, y, a, los, del, se, las, por, un, par, con*.[2]

SEGMENTACIÓN DE FRASES: Procedimiento que separa párrafos en sentencias. Presenta sus propios retos, ya que los signos de puntuación, como el punto (.) se usan con frecuencia para marcar tanto el fin de una frase como para denotar abreviaciones y números decimales.

POS TAGGING o etiquetado morfosintáctico. Paso que etiqueta cada palabra de una sentencia con su categoría morfosintática, como *adjetivo, nombre, verbo, adverbio* y *preposición*. Estas etiquetas pueden usarse como entrada para procesamientos futuros, como el parseo de

1 http://www.robotstxt.org/robotstxt.html
2 Para ver una lista completa de palabras visitar: http://snowball.tartarus.org/algorithms/spanish/stop.txt

dependencias (Objetivo de este trabajo) o como característica para el proceso de AA.

1.5 EL PIPELINE DE CORENLP

1.6 ESTADO DEL ARTE

Leer state-of-the-art de 1-s2.0-S1566253515000536-main y 1-s2.0-S1566253516301117

Parte II

OBJETIVOS DEL TRABAJO

En este apartado deberán aparecer con claridad los objetivos, inicialmente previstos en la propuesta de TFG, indicando el alcance para cada uno de ellos. Es conveniente indicar de manera precisa las interdependencias entre los distintos objetivos y también enlazarlos con los diferentes apartados de la memoria.

Los aspectos formativos previos más utilizados pueden ser destacados aquí.

Parte III

RESOLUCIÓN DEL TRABAJO

Se explicarán los métodos y procesos empleados para desarrollar el trabajo y alcanzar los objetivos. Es conveniente destacar tanto los métodos inicialmente previstos como aquellos que hayan tenido que ser agregados en el desarrollo del trabajo.

Éste es el lugar de presentar todos los datos técnicos y científicos realizados en el TFG. Debe ser detallado, claro y preciso.

En caso de ser un TFG en el que se desarrolle software, se recomienda que la sección 5 quede estructurada de la siguiente forma:

5.1 Planificación y presupuesto. Planificación temporal, con su correspondiente división en fases y tareas, y la posterior comparación con los datos reales obtenidos tras realizar el proyecto. También se incluir un presupuesto del trabajo a realizar.

5.2 Análisis y diseño. Se incluirá la especificación de requerimientos y la metodología de desarrollo por la que se ha optado, así como los "planos" del proyecto, que contendrán las historias de usuario o casos de uso, diagrama conceptual, diagramas de iteración, diagramas de diseño, esquema arquitectónico y bocetos de las interfaces de usuario. Además, se describirán las estructuras de datos fundamentales y los desarrollos algorítmicos no triviales.

5.3. Implementación y pruebas. En esta sección se incluirán todos los aspectos relacionados con la programación de la aplicación y las tecnologías seleccionadas, justificándolas e incluyendo el diseño de pruebas e informes de ejecución de las mismas.

Parte IV

CONCLUSIONES Y VÍAS FUTURAS

Las conclusiones deben incluir todas aquellas de tipo profesional y académico. Además, se debe indicar si los objetivos han sido alcanzados totalmente, parcialmente o no alcanzados.

Si hubiese claramente posibles vías de desarrollo posterior es interesante destacarlas, poniéndolas en valor en el contexto inicial del trabajo.

Parte V

APPENDIX

BIBLIOGRAFÍA

[1] Dong Yu George Dahl Abdel-rahman Mohamed Navdeep Jaitly Andrew Senior Vincent Vanhoucke Patrick Nguyen Brian Kingsbury Tara Sainath Geoffrey Hinton Li Deng. «Deep Neural Networks for Acoustic Modeling in Speech Recognition». En: *IEEE Signal Processing Magazine* 29 (2012), págs. 82-97. URL: https://www.microsoft.com/en-us/research/publication/deep-neural-networks-for-acoustic-modeling-in-speech-recognition/.

[2] J. Hirschberg y C. D. Manning. «Advances in natural language processing». En: *Science* 349.6245 (2015), págs. 261-266. DOI: 10.1126/science.aaa8685. URL: http://dx.doi.org/10.1126/science.aaa8685.

[3] Donald E. Knuth. «Computer Programming as an Art». En: *Communications of the ACM* 17.12 (1974), págs. 667-673.

[4] B. Liu. En: *Handbook of Natural Language Processing Chapter Sentiment Analysis and Subjectivity* (2010), págs. 627-666. URL: www.scopus.com.

[5] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard y David McClosky. «The Stanford CoreNLP Natural Language Processing Toolkit». En: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, págs. 55-60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[6] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard y David McClosky. «The Stanford CoreNLP Natural Language Processing Toolkit». En: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics (ACL), 2014. DOI: 10.3115/v1/p14-5010. URL: http://dx.doi.org/10.3115/v1/P14-5010.

[7] Bo Pang, Lillian Lee y Shivakumar Vaithyanathan. «Thumbs Up?: Sentiment Classification Using Machine Learning Techniques». En: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, págs. 79-86. DOI: 10.3115/1118693.1118704. URL: http://dx.doi.org/10.3115/1118693.1118704.

[8] Mike Pound y Sean Riley. *Inside a Neural Network – Computerphile*. Youtube. 2016. URL: https://www.youtube.com/watch?v=BFdMrDOx_CM.

[9] Hao Wang y Martin Ester. «A Sentiment-aligned Topic Model for Product Aspect Rating Prediction». En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics (ACL), 2014. DOI: 10.3115/v1/d14-1126. URL: http://dx.doi.org/10.3115/v1/D14-1126.

DECLARATION

Put your declaration here.

Granada, 20 de noviembre de 2016

Alejandro Alcalde Barros

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL*) are used. The "typewriter" text is typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

NOTE: The custom size of the textblock was calculated using the directions given by Mr. Bringhurst (pages 26–29 and 175/176). 10 pt Palatino needs 133.21 pt for the string "abcdefghijklmnopqrstuvwxyz". This yields a good line length between 24–26 pc (288–312 pt). Using a "*double square textblock*" with a 1:2 ratio this results in a textblock of 312:624 pt (which includes the headline in this design). A good alternative would be the "*golden section textblock*" with a ratio of 1:1.62, here 312:505.44 pt. For comparison, `DIV9` of the `typearea` package results in a line length of 389 pt (32.4 pc), which is by far too long. However, this information will only be of interest for hardcore pseudo-typographers like me.

To make your own calculations, use the following commands and look up the corresponding lengths in the book:

```
\settowidth{\abcd}{abcdefghijklmnopqrstuvwxyz}
\the\abcd\ % prints the value of the length
```

Please see the file `classicthesis.sty` for some precalculated values for Palatino and Minion.

145.86469pt