FIB UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Integración de conocimiento morfológico en un sistema de traducción estadístico chino-castellano.

Trabajo de final de grado

Autor: Carlos Escolano

Directores: Marta Ruiz Costa-jussà, Lluís Padró Cirera

Especialidad: Computación

29 de junio de 2016

Agradecimientos

Me gustaría agradecer a Marta Ruiz Costa-Jussà y a Lluis Padró por su ayuda, proporcionado los datos utilizados en el artículo original para la realización de los experimentos y por haber resuelto las posibles dudas y proporcionar consejos para llevar a buen puerto este desarrollo.

También merecen estos agradecimientos mi familia, amigos y compañeros de trabajo por su apoyo y su paciencia en estos meses en los que han oído hablar de traducción y aprendizaje automático más de los que les hubiera gustado.

Resumen

Productos como *Google Translate* nos han demostrado que existe una necesidad entre los usuarios de traducir textos en otros idiomas de una forma sencilla y transparente. El motivo de esta situación es que vivimos en una sociedad cada vez más globalizada en la que tenemos más contacto con culturas que se encuentran a miles de kilómetros de nosotros.

El objetivo principal de este proyecto es el desarrollo de una arquitectura que nos permita realizar una traducción de chino a castellano a partir de una traducción a castellano simplificado, al que se le ha eliminado la información morfológica[Costa-jussà, 2015] para posteriormente recuperar esta información y generar una traducción completa.

Como objetivo secundario, desarrollaremos un modelo de clasificador que nos permita para cada palabra recuperar su información, y para ello analizaremos distintas técnicas de aprendizaje automático y su comportamiento en la tarea que nos ocupa.

Además, para aplicar estas técnicas, utilizaremos diversos aspectos de procesado del lenguaje natural (*NLP*) que nos permitirán que la información relevante del texto a traducir pueda ser interpretada por los modelos generados.

Finalmente presentaremos las técnicas de evaluación humana y automática empleadas para evaluar los resultados obtenidos respecto a un sistema de referencia.

Palabras Clave: traducción automática, aprendizaje automático, procesado del lenguaje natural, redes neuronales convolucionales, redes neuronales recurrentes

Resum

Productes com *Google Translate* ens han demostrar que existeix una necessitat entre els usuaris de traduir textos en altres idiomes d'una forma senzilla y transparent. El motiu d'aquesta situació es que vivim en una societat cada cop més globalitzada on tenim cada vegada més contacte amb cultures que es troben a milers de quilòmetres de nosaltres.

L'objectiu principal d'aquest projecte és el desenvolupament d'una arquitectura de traducció xinés-castellà a partir d'una traducció a castellà simplificat, al qual s'ha eliminat la informació morfològica [Costa-jussà, 2015] per posteriorment recuperar aquesta informació y generar una traducció completa.

Com a objectiu secundari, desenvoluparem un model de classificador que ens permeti per a cadascuna de les paraules del text recuperar la seva informació, y per fer-ho analitzarem diferents tècniques d'aprenentatge automàtic y el seu comportament a la tasca que ens ocupa.

A més, per aplicar aquestes tècniques, utilitzarem diferents aspectes de processar del llenguatge natural (NLP) que ens permetran que la informació rellevant del text a traduir pugui esser interpretada per als models generats.

Finalment presentarem les tècniques d'avaluació humana y automàtica emprades per avaluar els resultats obtinguts respecte a un sistema de referència.

Paraules Clau: traducció automàtica, aprenentatge automàtic, processament del llenguatge natural, xarxes neuronals convolucionals, xarxes neuronals recurrents

Abstract

Products such as *Google Translate* have proved the existence of a necessity between users of translating text in other languages in an easy and transparent way. The main reason of this situation is that we are living in a society that every day is more globalized and where we have more contact with cultures that are from thousands of kilometers away.
The main objective of this project is the development of an architecture that allows us to translate from Chinese to Spanish from a simplified Spanish translation, from whom we have removed the morphological information [Costa-jussà, 2015] for later recovering this information and generating a full translation.

As a secondary objective, we are developing a model of classifier that allows us to get back this information. To do this we are going to analyze different machine learning techniques and their behavior in our current task.

In addition, to apply these techniques we are using different aspects of natural language processing to make arrive all of the relevant information from the texts to translate to the generated models.

Finally, we will present the evaluation techniques, both automatic and human based, to measure the results obtained compared to the base system.

Key Words: automatic translation, machine learning, natural language processing, convolutional neural networks, recurrent neural networks.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El campo del procesado del lenguaje natural (*NLP*) es un campo que está viviendo un rápido crecimiento y en el que constantemente vemos como se alcanzan nuevos resultados que anteriormente parecían imposibles. Su radio de acción se extiende desde la traducción que tratamos en este proyecto hasta la interpretación de fragmentos de sonido, como los que hacen posible la existencia de *Siri* en los dispositivos *Apple*.

De la misma forma el campo del aprendizaje automático también es un tema de gran interés entre la comunidad académica, gracias a la potencia de hardware de la que disponemos en la actualidad. Cada día vemos como son desarrolladas nuevas aplicaciones de técnicas como el *deep learning* en el mundo empresarial para analizar grandes volúmenes de datos.

Personalmente, el motivo para decidir llevar a cabo este proyecto fue la oportunidad de trabajar estas dos disciplinas. Generalmente pensamos en el aprendizaje automático aplicado a mediciones numéricas, pero en el caso que nos ocupa además trabajamos la selección de características que utilizaremos con nuestros modelos. Por todo ello este proyecto suponía afrontar estos retos desde un punto de vista diferente del visto anteriormente durante la carrera.

1.2. Alcance del proyecto

A continuación explicaremos los objetivos que buscamos completar con la realización del proyecto, su alcance y la metodología empleada en su desarrollo.

Objetivos

El objetivo de este proyecto es mediante la experimentación con distintos algoritmos de aprendizaje automático crear un modelo que permita añadir conocimiento morfológico a una salida

de traducción simplificada. La información obtenida con este modelo o modelos permite generar una etiqueta gramatical(*part-of-speech tag*) completa que podemos utilizar en un analizador de lenguaje para encontrar la flexión correcta de la palabra en la traducción. Con las palabras generadas y el alineamiento de éstas creado por el traductor del que partimos podemos generar una traducción completa y evaluar la precisión del modelo. Paralelamente a la experimentación con distintos algoritmos podemos realizar un estudio de sus resultados para la causa que nos ocupa.

La motivación para realizar esta tarea es, son los resultados obtenidos en el artículo *Ongoing Study for Enhancing Chinese-Spanish Translation with Morphology Strategies*[Costa-jussà, 2015] en los que de demostraba que al simplificar la información morfológica en la traducción mejoraban los resultados-

Como datos para realizar todo el proceso disponemos del corpus de datos utilizado en el artículo original, el cual consiste en un conjunto de textos procedentes de discursos de la ONU[Rafalovitch et al., 2009] de aproximadamente 60000 oraciones o 2.5 millones de palabras . Este conjunto es particularmente útil ya que disponemos de todos los textos con sus traducciones a chino y castellano en paralelo.

Hemos centrado toda la investigación en traducción chino-castellano pero realmente al no basarnos en reglas propias de las lenguas, los mismo mismos modelos pueden ser aplicados para otros idiomas únicamente ajustando sus parámetros, por ejemplo número de elementos que caracterizan una palabra del texto, etc. De hecho se ha participado con éxito en el *ACL 2016 first conference on machine translation (wmt16)*[Costa-jussà et al., 2016] creando un sistema de traducción inglés-castellano utilizando un corpus paralelo sobre biomedicina.

Alcance

Dentro de todo el proceso de traducción, en este proyecto nos centraremos exclusivamente en la tarea de generación de etiquetas gramaticales y en el post-procesado de éstas para generar una traducción. Concretamente, partiremos de una etiqueta en la cual se ha omitido la información que no se muestra explícitamente en el idioma. En el caso del chino esta información es el número y el género de la palabra.

Para lograr este objetivo necesitaremos en primer lugar seleccionar un método de pre-procesado de los datos el cual permita entrenar los modelos utilizados .

En segundo lugar debemos generar y evaluar los modelos obtenidos y seleccionar el modelo final que presente la mayor precisión en la tarea de clasificación. Para ello evaluaremos el comportamiento de diversos algoritmos sobre un conjunto de evaluación independiente.

Finalmente utilizando herramientas de análisis de lenguaje crear un texto final traducido y comprobar como de acertado es el resultado respecto del texto correctamente traducido.

Actores implicados

A continuación definimos los principales actores implicados en la realización del proyecto y en el resultado final.

- **Desarrollador:** La labor de desarrollo consiste buscar información para profundizar mi conocimiento sobre el dominio a tratar, buscar librerías y soluciones aplicables al proyecto e implementarlas.

- **Directores:** Los directores de este proyecto son Marta Ruiz Costa-Jussà y Lluís Padró Cirera. Durante la realización del trabajo han realizado una labor de guiar en el desarrollo, y marcar pautas para la correcta realización del mismo.

- **Usuario:** Podemos distinguir dos usuarios principales del proyecto. Aquellos que buscan traducir un texto de forma que todo el proceso interno sea transparente y aquellos que tienen como objetivo realizar su propio procesado de texto y puedan beneficiarse de las etiquetas generadas. Dentro del segundo grupo también podemos encontrar empresas que busquen realizar un post-procesado tras ejecutar un modelo de traducción

Metodología y rigor

En la presente sección describimos como las diferentes fases del proyecto han sido organizadas y los métodos utilizados para medir la precisión de los resultados obtenidos.

- **Pre-procesado:** En la primera fase estudiamos las características morfológicas tanto del idioma fuente como el idioma destino y decidimos como caracterizaremos las palabras del texto para que puedan ser procesadas por los algoritmos de clasificación.

- **Clasificación:** Durante esta fase del proyecto experimentamos con distintos métodos de clasificación y analizamos los resultados obtenidos con el objetivo de elegir aquel que se ajusta mejor a la tarea. Para ello utilizamos una metodología de desarrollo basado en prototipos.

- **Post-procesado:** Tras seleccionar el modelo que mejor se ajusta a los datos generamos el texto final traducido. Para ello completamos las etiquetas simplificadas de las que partíamos con los resultados de la fase anterior y utilizando un sistema de análisis de lenguaje para obtener la palabra completamente conjugada.

Desarrollo basado en prototipos

Esta metodología consiste en generar frecuentemente versiones que proporcionen valor añadido respecto a versiones anteriores y evaluar el efecto de los cambios implementados.

El principal beneficio que nos reporta este diseño es facilitar la tarea de analizar individualmente el impacto de las modificaciones realizadas y de la misma forma en caso de no obtener los resultados esperados recuperar una versión anterior.

Otra ventaja que nos proporciona este método es que a diferencia de otros planteamientos como por ejemplo el desarrollo en cascada no hemos de realizar una iteración completa en el proceso de desarrollo si queremos modificar alguna decisión tomada en una fase anterior. Esto resulta muy conveniente ya que diferentes algoritmos pueden requerir diferentes representaciones del texto de entrada.

Evaluación de los resultados

En las fases anteriormente mencionadas encontramos dos puntos en los que debemos utilizar métodos de evaluación para guiarnos en la toma de decisiones. Estos métodos son siempre evaluados utilizando un corpus de 1000 oraciones que no ha formado parte del entrenamiento de los modelos.

En primer lugar necesitamos una medida que nos permita seleccionar el algoritmo que nos proporciona los resultados más ajustados en la tarea de clasificación. Para ello utilizamos como medida de precisión el porcentaje de palabras correctamente clasificadas sobre el conjunto de validación del que conocemos previamente su resultado.

En segundo lugar, una vez terminada la fase de clasificación hemos de realizar las tareas de generación del texto traducido y su post-procesado. En esta fase podemos medir la calidad final de la solución creada en comparación con una traducción correcta. El método elegido para ello es el índice *METEOR* [Denkowski and Lavie, 2014] que nos permite a partir de un texto de referencia y un texto hipótesis (el texto generado) comparar su similitud y evaluarla en una escala de 0 a 100 donde 0 son dos textos completamente diferentes y 100 son dos textos idénticos. El motivo de elegir este método de validación es el hecho de que diversos experimentos han mostrado correlación entre el resultado obtenido por *METEOR* y el veredicto obtenido por un juez humano conocedor de los lenguajes[Banerjee and Lavie, 2005].

Además, como medida alternativa de la calidad de la solución proponemos un sistema de traducción humana, en el cual podemos observar si la arquitectura propuesta presenta ventajas para hablantes reales del idioma.

Control de versiones

En todo proyecto de *software* que se prolonga en el tiempo necesitamos disponer de algún sistema que nos permita disponer de copias actualizadas en caso de contingencias.

Además los sistemas de control de versiones nos proporcionan mecanismos para recuperar estados anteriores del desarrollo de una forma sencilla y sistemática. En nuestro desarrollo realizamos rápidas iteraciones de modificación y evaluación y disponer de un medio para revertir los cambios realizados cuando no proporcionan una mejora ahorra una gran cantidad de tiempo de depuración y de implementación de aspectos desarrollados anteriormente.

Además, al realizar el desarrollo entre distintas máquinas era necesario disponer de mecanismos que nos permitieran disponer de la versión más actualizada de nuestros datos entre ellas para garantizar que realizábamos los experimentos incluyendo los últimos cambios realizados.

Estudiamos diversas alternativas para realizar control de versiones. De entre ellas los factores que buscábamos en ellas era la posibilidad de administrar el código remotamente, para mantener siempre una copia en caso de fallos de *hardware* y que nos permitiera crear repositorios privados.

Finalmente escogimos *Bitbucket* por ser un sistema que utiliza *git* para realizar el proceso, nos permite disponer siempre de una versión alojada en sus servidores y proporciona la posibilidad de crear repositorios privados de forma gratuita.

Posibles riesgos

Diversas situaciones pueden comprometer que un desarrollo de software pueda ser entregado dentro de los plazos planeados inicialmente. A continuación listaremos las causas que hemos tenido en cuenta en este desarrollo.

En primer lugar, siempre estamos expuestos a fallos de hardware imprevistos que puedan incapacitarnos para continuar el desarrollo por un periodo de tiempo o incluso provocar la perdida de partes del proyecto ya desarrolladas. Para minimizar las consecuencias de estas situaciones utilizamos sistemas de control de versiones, gracias a las cuales seamos capaces de disponer siempre de una copia actualizada y que de forma sencilla podamos retomar el desarrollo en otra máquina, de ser necesario.

En segundo lugar, es común a la hora de realizar modificaciones en el código que bien por un error en el código o en la aplicación de las librerías y técnicas utilizadas aparezcan resultados no esperados o fallos en la ejecución. La depuración de estas situaciones en partes críticas del sistema puede provocar que alguna de las fases planificadas tenga que ser pospuesta.

Otro factor a tener en cuenta es el hecho de subestimar el esfuerzo necesario para realizar las tareas del proyecto. La dedicación necesaria para una tarea puede variar dependiendo de los resultados obtenidos al evaluarla una vez desarrollada. Para minimizar el impacto de este

factor reservamos un periodo tiempo del proyecto para evaluación como margen para realizar las modificaciones necesarias.

Y finalmente, empleamos técnicas que necesitan un periodo de entrenamiento que puede prolongarse durante días. Es importante para poder finalizar las tareas dentro de los plazos organizar la forma en la que las realizamos para evitar quedar bloqueados por precedencias pendientes de finalizar.

1.3. Contexto

En esta sección pretendemos exponer detalladamente el dominio del problema que pretendemos abordar, el estado del arte y las técnicas utilizadas en la realización del proyecto.

Contextualización

El objetivo de todo sistema de traducción es a partir de una entrada en un idioma generar una salida en otro idioma en la que se preserva lo más acertadamente posible el significado original.

En el proceso de traducción hemos de tener en cuenta tres factores principales.

- El primer lugar la semántica, el objetivo es que nos permitan relacionar palabras con las palabras del otro idioma que tengan el significado más cercanos a ellas posible.

- Sintaxis, queremos que la ordenación de las palabras en las oraciones traducidas sea coherente con la estructura que espera un hablante del idioma.

- Morfología, la estructura de la palabra y los motivos y reglas que la definen.

El reto al que nos enfrentamos al realizar una tarea de traducción es que los lenguajes que utilizamos para comunicarnos pertenecen al denominado lenguaje natural. Este conjunto de lenguajes se desarrolló de forma autónoma durante el paso del tiempo al ser utilizado por una comunidad de hablantes. Este proceso sucede sin ningún tipo de planificación y se produce una evolución constante que con el uso y las generaciones puede producir variaciones tan significativas que den origen a una nueva lengua.

Fruto de esta evolución natural se presentan características que dificultan su tratamiento por sistemas automáticos. Estos lenguajes presentan ambigüedades a la hora de ser interpretados, ya que el significado de una oración está determinado por el tono en el que ha sido pronunciada y por el contexto en el que sucede.

Cuanto más próximos sean los idiomas más sencilla será la tarea de generar una traducción y mejores serán los resultados. Un ejemplo de idiomas muy próximos son las lenguas latinas. Estas lenguas derivan del latín, lo que supone que sus oraciones se estructuran de forma similar y una gran parte de sus palabras tienen origen común.

Por el contrario en el caso del chino y el castellano encontramos importantes diferencias en su morfología que dificultan la traducción.

En primer lugar en el idioma chino diferencias tonales en una palabra modifican su significado [Li and Cheng, 1994], siendo los cuales imposibles de reflejar por escrito.Estos matices en la tonalidad no están presentes en el castellano, donde son representados en la morfología de la palabra. En segundo lugar ambos idiomas reflejan de forma muy diferente la información morfológica de las palabras. En el idioma chino esta información no esta reflejada directamente en el texto, sino que es el hablante que ha de deducirla a partir del contexto. En cambio el castellano sí que presenta explícitamente estas variaciones [Española, 2009]. Un ejemplo de esta situación es palabra Māo(en pinyin simplificado) la cual fuera de contexto puede ser traducida en castellano como gato, gata, gatos, gatas.

Como hemos comentado en apartados anteriores en un trabajo anterior [Costa-jussà, 2015] tras realizar diversos experimentos se concluyó que se obtenían mejores resultados al realizar una traducción a un conjunto simplificado y añadiendo la información morfológica posteriormente.

Principales áreas de interés

En la siguiente sección pretendemos hacer una breve introducción de las principales áreas de estudio relacionadas con el proyecto.

Aprendizaje automático

El aprendizaje automático (o *machine learning*) engloba un conjunto de técnicas que permiten realizar tareas de clasificación o regresión sobre unos datos mediante el uso de la estadística.

Todos estos algoritmos tienen en común que han de ser entrenados. Cuando hablamos de entrenamiento nos referimos a que estos algoritmos disponen de mecanismos por los cuales pueden ajustar sus parámetros al tratar un volumen de datos. Esta fase de entrenamiento puede ser supervisada, en la cual conocemos el resultado correcto y el algoritmo utiliza esa información para mejorar sus resultados, o no supervisado, donde no disponemos del valor correcto y podemos utilizar los resultados obtenidos para descubrir nuevas relaciones entre nuestros datos.
Podemos distinguir tres grupos principales de métodos con los que experimentaremos en este proyecto:

- **métodos lineales:** Estos métodos permiten separar mediante una función lineal los valores que buscamos predecir. Ejemplos de ello puede ser realizar regresión lineal sobre unos datos para predecir su valor en el futuro

- **métodos no lineales:** En casos en que las fronteras entre los valores que buscamos predecir no se ajuste a una recta los modelos lineales no pueden aproximar correctamente nuestros datos. Con el uso de métodos no lineales buscamos por medio calcular con nuestro

modelo una función no lineal (polinomio, tangente hiperbólica, etc.) poder modelar esa varianza presente en los datos. Podemos destacar entre estos métodos los basados en *support vector machines*, los cuales se basan en dar más importancia en el modelado a los puntos que están más cercanos a las fronteras entre las clases, y las redes neuronales, las cuales se basan en una o varias capas por las que los datos son tratados con una función no lineal y un conjunto de pesos entre las capas.

- **métodos de *ensemble*:** Esta familia de métodos se fundamenta en un mecanismo para generar submuestras (por ejemplo *boosting*), entrenar un conjunto de modelos con estas submuestras y combinar sus salidas para generar un resultado final. Un ejemplo de esta técnica es *random forest* donde utilizamos un conjunto de árboles.

Sistemas de traducción.

En cuanto a traducción se refiere podemos distinguir dos tipos de aproximaciones desde puntos de vista diferentes.

En primer lugar podemos agrupar los traductores basados en reglas en los que un grupo de lingüistas han de generar manualmente un conjunto de reglas que caractericen como trasladar las estructuras propias de un idioma a otro. El principal inconveniente de esta técnica es que requiere de un trabajo laborioso por parte de expertos en lingüística, los cuales han de disponer de un amplio conocimiento tanto de la lengua fuente como de la lengua destino.

El otro planteamiento, en el que basa este proyecto, consiste en entrenar algoritmos de aprendizaje automático con características obtenidas a partir de textos en los idiomas a tratar. Estas características pueden ser generadas a partir del conocimiento sobre lingüística del personal en el proyecto o puede ser generado mediante el uso de algoritmos no supervisado, como puede el caso de utilizar *.embeddings*[1] para generar la entrada que recibirá el modelo.

Existe una tercera vía [Costa-jussà, 2015] que consiste en crear un método híbrido que combine técnicas de los dos sistemas anteriores.

En el caso del proyecto que nos ocupa partimos de un modelo ya existente el cual genera en lugar de una palabra en castellano un conjunto de lema más etiqueta gramatical simplificada. [Costa-jussà, 2015] Definimos como lema la raíz una palabra de un idioma, aquella que de la cual podemos generar el resto de flexiones añadiendo afijos (prefijos, infinitos y sufijos).

Utilizamos el término etiqueta gramatical, o *part-of-speech tag*, para referirnos a la representación para una palabra de su información gramatical. Ejemplos de información que encontramos son el tipo de palabra, en el caso de verbos tiempo y persona o distinguir si un determinante es definido o indefinido.

1.4. Estado del arte.

Diversos estudios han sido realizados con el objetivo de realizar sistemas de traducción cada vez más precisos. En esta sección realizaremos en breve resumen de los resultados obtenidos en este campo.

Los sistemas de traducción automática han sido estudiados durante largo tiempo, pero no es hasta la década de 1990 en que surgen los sistemas de traducción estadísticos, gracias a los nuevos algoritmos propuestos, por ejemplo el uso de funciones de *kernel* en *support vector machines* propuesto en 1992 [Boser et al., 1992], y a los avances en la potencia de cálculo disponible.

En el caso de la asignación de etiquetas gramaticales diversos estudios se han realizado durante el transcurso de los años destacando los resultados obtenidos para el inglés, alcanzando un 97.24 % de acierto en 2003 o los alcanzados en 2007 con un 97.33 % [Collobert et al., 2011].

[1] Como veremos en el apartado de base teórica.

Sin embargo encontramos que la precisión obtenida con modelos basados en el chino es notablemente inferior. Los modelos que han demostrado un porcentaje de acierto mayor son aquellos basados en analizar las palabras a nivel de carácter con resultados de 93,8 % y 94,01 % [Shen et al., 2014]. En contraste con los modelos orientados al inglés en los que se analizaban palabras completas del texto.

Frontera de conocimiento

Como hemos visto en el apartado anterior encontramos cierto margen de mejora entre los resultados obtenidos entre lenguas europeas y los que obtenidos entre al realizar traducciones del chino.

A su vez vemos como en los últimos tiempos ha surgido un gran interés por las redes neuronales y el *deep learning* que han demostrado ser muy eficaces en tareas de clasificación como pueden ser reconocimiento de imagen o incluso análisis de críticas de cine [Socher et al., 2013].

1.5. Uso de resultados anteriores

En el transcurso de este proyecto hemos utilizado un conjunto de herramientas ya desarrolladas, las cuales nos han facilitado la tarea de desarrollo y han aportado recursos cuya creación esta fuera del alcance de este proyecto.

- ***Moses*** [Koehn et al., 2007] : Sistema de traducción estadístico que permite crear un modelo de traducción desde cero. En este proyecto utilizaremos las herramientas que incluye para preprocesado de texto y para generar un modelo de traducción base sobre el cual poder evaluar los resultados obtenidos.

- ***srilm*** [Stolcke, 2002] : Herramienta que permite crear modelos estadísticos de lenguaje.Nos permite crear un modelo de lenguaje, necesario para el modelo de traducción de *Moses*.

- ***Stanford parser*** [Chen and Manning, 2014] : *Suite* estadística de procesado de lenguaje desarrollada por la universidad de Stanford. Nos permite etiquetar y encontrar dependencias sobre el corpus de texto en chino.

- ***Freeling*** [Padró and Stanilovsky, 2012, Padró, 2011, Padró et al., 2010]

  [Atserias et al., 2006, Carreras et al., 2004]: Suite de herramientas de análisis de lenguaje desarrollada por el grupo de investigación sobre lenguaje natural y dirigida por Lluís Padró. Para la realización de este proyecto la utilizamos en la fase de pre-proceso para separar contracciones presentes en el texto original y encontrar dependencia entre las palabras de las oraciones y en la fase de post-procesado ya que a partir de una combinación lema etiqueta nos permite obtener la flexión de la palabra.

- ***scikit-learn*** [Pedregosa et al., 2011] : Librería para *python* dedicada al aprendizaje automático que proporciona la implementación y optimización de una gran variedad de algoritmos supervisados y no supervisados que utilizaremos durante la experimentación en la fase de procesado del proyecto.

- ***Keras*** [Chollet, 2016] : Librería para *python* que nos proporciona una capa de abstracción a la hora de crear modelos utilizando redes neuronales y *deep learning*. La experimentación con redes neuronales de este proyecto será realizada con ella.

1.6. Atribuciones

En esta sección pretendemos resumir las fases del proyecto que van a ser implementadas específicamente para el desarrollo de nuestro proyecto.

En trabajos anteriores se han presentado arquitecturas para la generación de etiquetas gramaticales, que es la labor más cercana a la tarea que nos proponemos, pero a diferencia de estos trabajos nosotros solo pretendemos generar elementos concretos de esta etiqueta a partir de una ya existente.

Debido a ello para conseguir resultados cercanos a los presentados en el estado del arte hemos de adaptar las arquitecturas existentes para las necesidades de nuestra tarea. Durante el proyecto creamos diversos *scripts*, así como la arquitectura del clasificar únicamente para nuestra tarea.

De la misma forma el post-procesado del texto presenta la dificultad de completar las etiquetas y combinándolas con el lema de la palabra encontrar la flexión que más se acerca a la traducción correcta.

1.7. Estructura de la memoria

En esta sección presentaremos los distintos capítulos en los que hemos estructurado este documento:

- *Planificación del proyecto:* Describimos las distintas fases en las que hemos organizado el desarrollo y su coste financiero y temporal.

- *Ejemplo:* Presentamos el proceso para traducir una oración utilizando nuestro sistema a través de un caso concreto.

- *Base teórica:* Explicamos los distintos conceptos sobre *procesado del lenguaje natural* y aprendizaje automático empleados en el desarrollo.

- *Análisis de los datos:* Presentamos el conjunto de datos utilizado para entrenar y evaluar el sistema y describimos sus características.

- *Arquitectura:* Describimos las diferentes fases de nuestra arquitectura, explicando su función concreta para realizar la tarea de traducción.

- *Fuentes de información y métodos de clasificación:* En esta sección presentamos enfoques alternativos a la arquitectura propuesta, con los cuales experimentamos hasta definir el modelo final.

- *Resultados:* Presentamos los resultados obtenidos por nuestra arquitectura sobre un conjunto de evaluación comparados con los obtenidos por un sistema de referencia.

- *Conclusiones:* Presentamos los objetivos cumplidos en el desarrollo del proyecto y posible trabajo a realizar en el futuro.

Capítulo 2

Planificación del proyecto

A continuación presentamos las distintas tareas en las que se ha estructurado este proyecto, en que consisten y sus relaciones de precedencia.

Planificación inicial.

Antes de poder iniciar el desarrollo debíamos profundizar en nuestro conocimiento sobre el dominio a tratar, la traducción y más concretamente la traducción del idioma chino.

Para ello las semanas iniciales se recopilo información sobre gramática, tanto en chino como en castellano, sistemas de traducción y algoritmos de *machine learning* aplicados a la generación de etiquetas gramaticales.

Con todo ello realizamos una investigación sobre el estado del arte en la materia que nos permite determinar con que algoritmos experimentaremos durante la fase de desarrollo.

Además con la información obtenida podemos acotar el alcance de nuestro proyecto y comenzar a definir las tareas de las que se compondrá nuestro proyecto y planificar como las realizaremos en el tiempo.

Preparación del entorno de trabajo

La búsqueda de información del apartado anterior también nos ayuda a decidir que herramientas de software serán necesarias para la realización del proyecto.

En esta fase comprobamos que los recursos de hardware de los que disponemos son compatibles con las herramientas que hemos seleccionado y obtenemos las licencias de ellas, en caso de ser necesario. Para la realización de este proyecto a excepción de *Sublime Text 3* todo el software es software libre. *Sublime Text 3* es de pago pero puede ser utilizado gratuitamente.

Finalmente, instalamos las herramientas y resolvemos los problemas de dependencias en caso de encontrarlos.

Desarrollo y experimentación

En esta sección definimos las diferentes tareas en las cuales subdividimos el desarrollo de nuestro proyecto.

- *Pre-proceso:* En primer lugar hemos de decidir como representamos las palabras de nuestro texto para poder ser tratadas por los algoritmos.

- *Experimentación:* En esta fase probamos distintos algoritmos para seleccionar aquel que mejor predice nuestros datos. Podemos distinguir tres subfases en las cuales probamos cada una de las familias de algoritmos descritas en la contextualización(lineal, no lineal y *ensemble*).Existe una relación de precedencia de la tarea anterior, ya que su resultado será utilizado como entrada de nuestros algoritmos.

- *Post-proceso:* En esta fase, utilizando los resultados de la fase anterior generamos un texto final traducido. Existe la misma relación de precedencia del punto anterior, su entrada es el resultado de la fase de desarrollo.

- *Evaluación:* Finalmente hemos de evaluar los resultados de la solución propuesta comparados con los resultados obtenidos anteriormente.

Memoria y presentación

Durante todo el desarrollo hemos recabado información sobre los distintos aspectos trabajados y hemos mantenido un registro de los experimentos realizados. En esta última fase del desarrollo revisamos todos estos conceptos,matizando aquellos aspectos que podamos completar con la información que nos proporciona tener la arquitectura completa.

Con la estructura de la memoria decidida podemos comenzar a plantear como plantearemos la presentación oral del proyecto, que aspectos queremos enfatizar y como resumir el proceso realizado en un tiempo límitado de forma clara.

2.1. Asignación de horas

A continuación presentamos una relación de las tareas a realizar y las horas a dedicar a cada una de ellas.

Tabla 2.1: Relación tarea-horas

Tarea	Estimación de horas	Horas reales
Planificación	80	80
Pre-proceso	70	70
m. lineales	40	30
m. no lineales	90	100
m. ensemble	30	30
Post-proceso	60	100
Evaluación	90	90
Fase final	120	100
Total	580	**600**

Observamos diferencias entre las horas estimas y las horas dedicadas en realidad debido a que en la arquitectura final hemos realizado un paso adicional de post procesado no contemplado en la planificación inicial así como contrastar los resultados obtenidos por la evaluación automática con una evaluación humana.

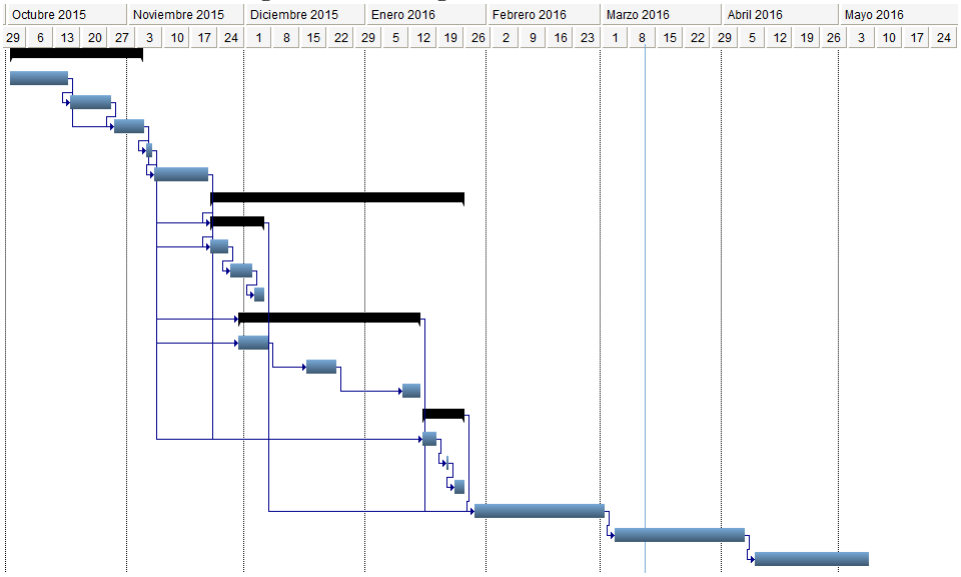
Diagrama de *Gantt*

En esta sección presentamos la asignación de horas a cada tarea y el diagrama de *Gantt* mostrando las relaciones de dependencia entre ellas.

Figura 2.1: Dedicación a las diferentes tareas

	Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
1	**Planificación**	**24d?**	**01/10/2015**	**03/11/2015**		
2	Contextualización	11d?	01/10/2015	15/10/2015		
3	Alcance	7d?	16/10/2015	26/10/2015	2	
4	Estado del arte	6d?	27/10/2015	03/11/2015	2,3	
5	Preparación del entorno	2d?	04/11/2015	05/11/2015	4	
6	Pre-proceso	10d?	06/11/2015	19/11/2015	4,5	
7	**Desarrollo y experimentación**	**46d?**	**20/11/2015**	**22/01/2016**		
8	**Métodos lineales**	**10d?**	**20/11/2015**	**03/12/2015**	**5,6**	
9	Desarrollo	3d?	20/11/2015	24/11/2015	5,6	
10	Ajuste	4d?	25/11/2015	30/11/2015	9	
11	Evaluación	3d?	01/12/2015	03/12/2015	10	
12	**Métodos no lineales**	**32d?**	**27/11/2015**	**11/01/2016**	**5,6**	
13	Desarrollo	6d?	27/11/2015	04/12/2015	5,6	
14	Ajuste	6d?	14/12/2015	21/12/2015	13	
15	Evaluación	3d?	07/01/2016	11/01/2016	14	
16	**Métodos de ensemble**	**9d?**	**12/01/2016**	**22/01/2016**		
17	Desarrollo	4d?	12/01/2016	15/01/2016	5,6	
18	Ajuste	1d?	18/01/2016	18/01/2016	17	
19	Evaluación	3d?	20/01/2016	22/01/2016	18	
20	Post-proceso	25d?	25/01/2016	26/02/2016	8,12,16	
21	Evaluación de los resultados	25d?	29/02/2016	01/04/2016	20	
22	Fase final	21d?	04/04/2016	02/05/2016	21	

Figura 2.2: Diagrama de *Gantt* de las diferentes tareas



Plan de acción

En esta sección pretendemos explicar como vamos a realizar el proyecto en los plazos especificados.

En primer lugar existen fuertes relaciones de dependencia entre las diferentes tareas. Es decir, sin tener un modelo preliminar de la fase de preproceso no podemos iniciar la fase de desarrollo, ya que su resultado sirve como entrada del desarrollo. Y de la misma manera no podemos realizar la fase de post-procesado sin encontrar modelos que se ajusten bien a nuestros datos.

En caso de retrasarnos en la realización del proyecto hemos reservado espacio en las fases de evaluación y fase final. En motivo de esta decisión es que son fases en las que repasamos todo el material creado y es posible que tengamos que realizar algún cambio ya sea para solucionar problemas o realizar mejoras.

Realizaremos reuniones semanales con la directora del proyecto con la intención de resolver dudas en el proceso y realizar un seguimiento del estado de la realización de las tareas.

Valoración de alternativas

La planificación de tareas propuesta puede variar por diversos motivos, como pueden ser fallos de software o factores externos que impidan dedicar el tiempo necesario para llevar las tareas a cabo.

Otro factor a destacar es el hecho de que el desarrollo se lleva a cabo mediante desarrollo basado en prototipos, lo cual promueve realizar varias iteraciones sobre las diferentes tareas a realizar en lugar de la cascada que muestra el diagrama de *Gantt*. A pesar de ello si es cierto que la estructura de estas iteraciones es la mostrada.

Recursos

En esta sección describiremos los recursos de hardware y software empleados en la realización del proyecto

Hardware

Para llevar a cabo este proyecto utilizaremos tres máquinas diferentes. Un ordenador portátil, por comodidad y para ejecutar tareas que no requieran gran potencia de cálculo. Un servidor que nos permite realizar tareas paralelamente utilizando su tarjeta gráfica. Y un servidor en el cual también podemos realizar tareas extensas en cálculo y que dispone de las herramientas de traducción necesarias. Ambos servidores son propiedad de la Universidad Politécnica de Cataluña(UPC) que nos ha dado acceso.

- Ordenador portátil:
  - *Sistema operativo*: *Ubuntu 14.04*
  - *Procesador*: *Intel I3 2 núcleos 1.5GHz*, tarjeta gráfica integrada
  - *Memoria*: 4GB DRR3
  - *Disco Duro*: 500GB HDD
- Servidor 1 (tarjeta gráfica):
  - *Sistema operativo*: *Fedora release 17*
  - *Procesador*: *Intel Xeon E5-2650* 8 núcleos
  - *Tarjeta gráfica*: *nVIDIA GeForce GTX TITAN*
  - *Memoria*: 100GB
- Servidor 2 (traducción):
  - *Sistema operativo*: *CentOS 10*
  - *Procesador*: *Intel Xeon 2.8GHz* 6 nucleos
  - *Memoria*: 100GB

Software

A continuación enumeramos el conjunto de software utilizado durante la realización del proyecy o, así como una breve descripción de en que fase se utiliza.

- *Ubuntu*: Durante todo el proyecto.
- *Fedora*: Durante la fase de desarrollo y experimentación.
- *CentOS*: Durante todo el proyecto.
- *Keras*[Chollet, 2016]: Durante la creación de los modelos.
- *Sckit-Learn*[Pedregosa et al., 2011]: Durante todo el proyecto.
- *Freeling*[Padró and Stanilovsky, 2012], [Padró, 2011], [Padró et al., 2010], [Atserias et al., 2006], [Carreras et al., 2004]: Durante todo el proyecto.
- *Moses*[Koehn et al., 2007]: Durante todo el proyecto.
- *Git*: Durante todo el proyecto.
- *Sublime Text 3*: Durante todo el proyecto.
- *Vim*: Durante todo el proyecto.
- *Latex*: Durante todo el proyecto.

2.2. Identificación de costes

En esta sección describiremos los distintos costes a afrontar durante la realización de este proyecto. Haremos especial mención a los costes relacionados con el personal involucrado en el proyecto, recursos de hardware, software, costes indirectos y posibles imprevistos o contingencias.

Personal

Durante la realización de un proyecto de software son necesarios una serie de roles, entre los cuales repartimos las distintas tareas para llevar a buen puerto el desarrollo. Destacamos en el caso que nos ocupa los siguientes:

- **Jefe de proyecto:** Su cometido es obtener la información necesaria para crear una estructura del proyecto y planificar las diferentes fases del desarrollo. También realiza las reuniones con los clientes y realiza la especificación del producto a partir de sus requisitos.

- **Diseñador:** A partir de la especificación crea un esquema de la estructura de la aplicación y de la persistencia de la información necesaria.

- **Programador:** Realiza la tarea de crear código funcional a partir de las abstracciones creadas en las fases anteriores.

- ***Tester:*** Su labor es la creación de casos de prueba que aseguren el correcto funcionamiento de la aplicación y reportar los resultados erróneos que encuentre en el proceso.

Aunque aparentemente estos roles parece que se suceden en distintas fases de la aplicación debido a que el desarrollo esta basado en prototipos estos roles se suceden en diversas iteraciones durante el desarrollo.

Costes indirectos

Hacer funcionar los sistemas anteriores supone una serie de gastos que también hemos de tener en cuenta a la hora de estimar el coste de nuestro proyecto.

En primer lugar, necesitamos electricidad para hacer funcionar los sistemas hardware anteriores. De la misma forma para descargar a los recursos de software y conectarnos a los servidores de forma remota necesitamos acceso internet y debido a que será necesario mover archivos de gran tamaño, necesitaremos más velocidad, lo que aumentará el coste.

Software

Diversos recursos de software ha sido utilizados durante la realización de este proyecto, tal como especificamos en el apartado de planificación.

Debido a la naturaleza de la tarea a realizar, traducción estadística, gran parte de este software ha sido realizado por universidades y publicado bajo licencias de software libre(GPL, MIT). Lo cual supone que permiten su uso gratuitamente siempre que citemos a los autores e incluso en algunos casos modificar su código para ajustarlo a nuestras necesidades.

Además, todo el desarrollo a sido realizado en sistemas Linux que no tienen coste y además nos permiten una interacción más sencilla con el resto de software utilizado.

Para el control de versiones hemos utilizado la plataforma *bitbucket* de *Atlassian* que nos permite tener un repositorio privado donde alojar el código sin coste.

Del conjunto de software empleado únicamente *sublime text 3* dispone de una versión de pago, pero para este proyecto hemos utilizado la gratuita.

Hardware

En la realización de este proyecto hemos utilizado exclusivamente las siguientes máquinas:

- PC portátil utilizado para el desarrollo y para acceder remotamente a los servidores.
- Servidor con GPU utilizado para entrenar los algoritmos durante el entrenamiento en menos tiempo
- Servidor de traducción: utilizado para las tareas de traducción ya que supone de un entorno configurado para ello y nos permite ejecutar procesos que pueden durar varios días que no serían posibles en nuestro pc.

2.3. Estimación de costes

En esta sección presentaremos los presupuestos estimados de los diferentes aspectos de la realización del proyecto. Para los cálculos realizados contamos que el 21 % de I.V.A ya están presentes en los importes proporcionados.

Presupuesto coste en personal

En la siguiente tabla mostramos los costes por hora de los diferentes roles implicados en el desarrollo, así como el número de horas empleadas:

Tabla 2.2: Coste por hora de los roles implicados en el proyecto

Rol	Coste/hora
Jefe de proyecto	30€
Diseñador/ Analista	25€
Programador / Tester	10€

También podemos calcular la distribución de horas y el coste de cada una de las tareas del diagrama de *Gantt* del apartado de planificación:

Tabla 2.3: Costes asociados a personal.

Tarea	Coste	Jefe de Proyecto	Diseñador/Analista	Programador	Tester
Planificación	2250,00€	50	30		
Preproceso	1010,00€	2	18	40	10
m. lineales	555,00€	1	9	20	10
m. no lineales	1430,00€	1	28	50	20
m. ensemble	455,00€	1	9	10	10
Postproceso	1320,00€	1	20	34	45
Evaluación	940,00€	2		29	59
Fase final	2600,00€	70		15	35
Total	10560,00€				

Presupuesto coste en software

Como hemos mencionado en apartados anteriores hemos utilizado únicamente software libre o gratuito durante el desarrollo, por lo que presupuestamos un coste 0 tal como muestra la siguiente tabla:

Tabla 2.4: Coste del software utilizado en el desarrollo

Producto	Coste	Vida Útil	Amortización
Ubuntu Linux	0.0€	3 años	0.0€
Fedora	0.0€	3 años	0.0€
CentOS	0.0€	3 años	0.0€
Freeling	0.0€	3 años	0.0€
Keras	0.0€	3 años	0.0€
sckit-learn	0.0€	3 años	0.0€
sublime text 3	0.0€	3 años	0.0€
LATEX	0.0€	3 años	0.0€
SRILM	0.0€	3 años	0.0€
Total	0.0€		

Presupuesto coste en hardware

La tabla a continuación muestra el coste del hardware empleado en el el proyecto:

Tabla 2.5: Hardware utilizado

Producto	Coste	Vida Útil	Amortización
PC	700,00€	4 años	87,50€
Servidor1	10000,00€	4 años	1250,00€
Servidor2	13000,00€	4 años	1625,00€
Total	23700,00€		2962,5€

Presupuesto costes indirectos

A continuación mostramos el coste estimado de los gastos indirectos:

Tabla 2.6: Costes indirectos

Producto	Coste
Electricidad	200
Acceso a internet	300
Total	500

Presupuesto total

En esta sección mostramos la estimación total del coste del proyecto:

Tabla 2.7: Control de desviaciones

Presupuesto	Coste
Personal	10560,00€
Software	0,00€
Hardware	23700,00€
Indirectos	500,00€
Contingencias	10 %
Total	38236,00€

2.4. Control de desviaciones

En el desarrollo de nuestro proyecto, al tratarse de rápidas iteraciones en las cuales creamos los prototipos, es posible que exista variaciones entre los costes estimamos y los que se produzcan realmente. Estas situaciones intentan limitarse al reservar una parte del tiempo de cada iteración a realizar pruebas sobre los prototipos para mediante test acabar de acotar aquellos puntos que no funcionen correctamente de la aplicación.

De la misma forma en la fase de evaluación realizamos un revisión de todo el código con el objetivo de mejorar los resultados o reparar problemas que hubiéramos arreglado anteriormente, ya que dispondremos de más información sobre el resultado final y podremos utilizar métricas para medir su calidad.

Reservamos un 10 % del presupuesto a posibles contingencias que puedan suceder durante el desarrollo como pueden ser fallos en hardware que supongan tener que realizar un desembolso imprevisto, o posible compra de libros o material necesario para planificar como afrontar el proyecto.

2.5. Planificación definitiva

En la realización del proyecto hemos encontrado diversas desviaciones sobre la valoración original, en su mayoría han podido ser resueltas dentro del plan establecido pero algunos nos han llevado a modificar los plazos iniciales.

En primer lugar encontrar una arquitectura que nos proporcionara resultados satisfactorios. Durante la fase de experimentación diversas arquitecturas fueron realizadas en búsqueda de resultados que mejoren al sistema de referencia. Gracias a realizar iteraciones muy cortas los modelos que no mostraban mejoras eran descartados rápidamente, permitiéndonos realizar la fase de experimentación dentro del plazo propuesto.

Sin embargo si que encontramos desviaciones de la planificación en la fase de evaluación y post proceso. Éstas se deben a que subestimamos en la planificación inicial de la dificultad de mejorar los resultados obtenidos en el sistema de referencia.

Al afrontar la tarea comprobamos que además de generar unos buenos modelos de clasificación era necesario generar un conjunto de reglas que de post procesado para limitar el impacto en los resultados de factores externos a la morfología que se reflejaban en la traducción. Un ejemplo de ello es la creación de reglas que modificaran las conjunciones frente a vocales.

Además una nueva técnica de reevaluación de las probabilidades obtenidas fue añadida al proyecto, la cual no contemplamos en la fase inicial.

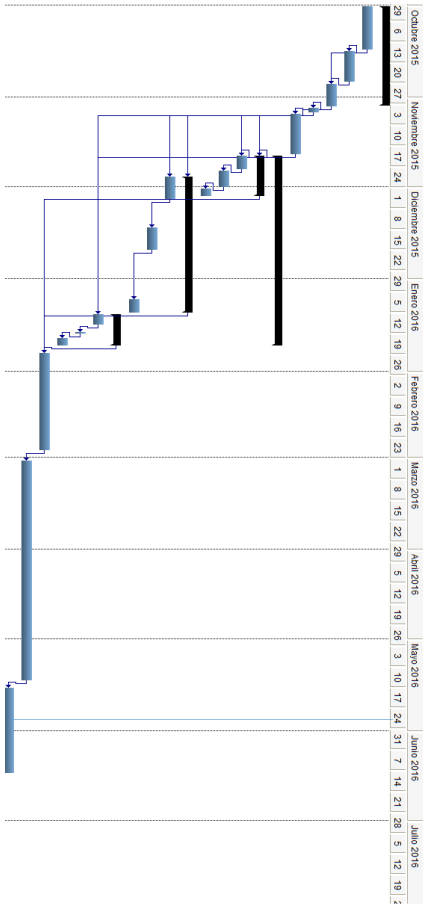
Pese a estas desviaciones, todos los procesos planificados han sido llevados a cabo, añadiendo el nuevo proceso de reevaluación en el post procesado.

Esto nos ha ocasionado un aumento en los costes de desarrollo, ya que hemos de hacer frente al coste de las horas trabajadas en esta nueva tarea así como de los costes indirectos del consumo eléctrico y de conexión a internet para realizarlos. Para hacerles frente recurrimos a la partida destinada a contingencias del presupuesto inicial.

Figura 2.3: Planificación final del proyecto

	Nombre	Duración	Inicio	Fin	Predecesoras
1	**Planificación**	**24d?**	**01/10/2015**	**03/11/2015**	
2	Contextualización	11d?	01/10/2015	15/10/2015	
3	Alcance	7d?	16/10/2015	26/10/2015	2
4	Estado del arte	6d?	27/10/2015	03/11/2015	2,3
5	Preparación del entorno	2d?	04/11/2015	05/11/2015	4
6	Pre-proceso	10d?	06/11/2015	19/11/2015	4,5
7	**Desarrollo y experimentación**	**46d?**	**20/11/2015**	**22/01/2016**	
8	**Métodos lineales**	**10d?**	**20/11/2015**	**03/12/2015**	**5,6**
9	Desarrollo	3d?	20/11/2015	24/11/2015	5,6
10	Ajuste	4d?	25/11/2015	30/11/2015	9
11	Evaluación	3d?	01/12/2015	03/12/2015	10
12	**Métodos no lineales**	**32d?**	**27/11/2015**	**11/01/2016**	**5,6**
13	Desarrollo	6d?	27/11/2015	04/12/2015	5,6
14	Ajuste	6d?	14/12/2015	21/12/2015	13
15	Evaluación	3d?	07/01/2016	11/01/2016	14
16	**Métodos de ensemble**	**9d?**	**12/01/2016**	**22/01/2016**	
17	Desarrollo	4d?	12/01/2016	15/01/2016	5,6
18	Ajuste	1d?	18/01/2016	18/01/2016	17
19	Evaluación	3d?	20/01/2016	22/01/2016	18
20	Post-proceso	25d?	25/01/2016	26/02/2016	8,12,16
21	Evaluación de los resultados	54d?	01/03/2016	13/05/2016	20
22	Fase final	21d?	16/05/2016	13/06/2016	21

Figura 2.4: Diagrama de Gannt definitivo del proyecto



Como se aprecia en el diagrama de Gantt de la planificación definitiva la fase de evaluación es la que muestra una mayor diferencia respecto a la planificación anterior.

2.6. Sostenibilidad

En esta sección trataremos sobre la sostenibilidad del proyecto centrándonos en tres aspectos de ésta: la Económica, Social y Ambiental. En la tabla mostramos las diferentes puntuaciones que otorgamos a este proyecto en estos apartados, que justificaremos en los apartados siguientes.

Tabla 2.8: Puntuaciones de los diferentes apartados de sostenibilidad

S. económica	S. social	S. ambiental
8	6	7

Sostenibilidad económica

Para la realización de este proyecto hemos pretendido maximizar los recursos disponibles e intentar reducir los costes.

En primer lugar, ya que gran parte del software que necesitábamos era gratuito intentamos utilizar siempre que fuera posible alternativas gratuitas, como por ejemplo elegir sublime text 3 como editor sobre otras opciones.

En segundo lugar, el presupuesto dedicado a los servidores. El hecho de utilizar servidores nos supone una serie de ventajas. Al permitirnos uno de ellos realizar calculos en paralelo en la GPU podemos reducir los coste indirectos de ejecutar los procesos en un pc.

A la vez, estos servidores no son únicamente dedicados a esta tarea, si no que siempre se mantienen encendidos ejecutando procesos de otros usuarios.

Otro punto a destacar es que su uso nos a evitado tener que realizar procesos que supongan tener nuestro pc durante días encendido ejecutando. Lo cual se refleja en menos contingencias relacionadas con fallos de la máquina y que al estar preparados para este tipo de ejecuciones tenemos mayor probabilidad de poder acabar los procesos sin incidencias.

Respecto a la viabilidad del proyecto, existen ya soluciones de traducción comerciales que han demostrado que es posible obtener beneficios de un sistema de estas características. Es un problema común para cualquier tipo de usuario encontrarse frente a un texto en un idioma desconocido y necesitar un sistema que pueda arrojar cierta luz sobre él.

De esta forma es posible en fases futuras, si el proyecto es mantenido, implantar la solución creada a un portal web y monetizarlo por medio de publicidad o subscripciones de pago.

Otra posibilidad de monetización es ofrecer el producto como un módulo que permita extender sistemas creados por empresas externas.

En conclusión, la traducción es un área en la que existe una fuerte demanda y que nos permite poder amortizar nuestra aplicación ya sea como producto dirigido al público o como extensión para software realizado por terceros. Por lo que valoramos este proyecto con un 8 en el apartado de mantenibilidad económica.

Sostenibilidad social

El usuario objetivo de nuestra aplicación es amplio, desde usuarios que buscan traducir un text o hasta empresas que busquen un módulo que permita traducir partes de sus propias aplicaciones.

Algunos usos que puede recibir esta aplicación es generar un sistema que permita a personas que se encuentren en un país extranjero, ya sea por turismo o por trabajo, poderse comunicar con los habitantes del lugar. Podríamos evitar con ello situaciones en las cuales una persona extranjera se encuentra perdida, o ha sido víctima de un delito y no puede comunicarse correctamente.

Otra situación en la que un sistema de traducción puede tener impacto social es en casos en los que se produce una catástrofe y es necesaria ayuda internacional. Tener un modo de traducir el idioma local en otro idioma puede resultar de gran ayuda. El sistema de traducción podría integrarse en una aplicación móvil y podría ser utilizado incluso en zonas sin acceso a internet.

Por lo tanto valoramos este proyecto con un 6 en lo relacionado con mejorar la vida de las personas, ya que pese a que trabajos futuros y extensiones pueden dar lugar a aplicaciones que verdad puedan suponer una diferencia, el proyecto por si mismo tiene un impacto reducido.

Sostenibilidad ambiental

Para realizar nuestro proyecto, durante la fase de entrenamiento de los algoritmos en la fase de experimentación necesitamos mantener una máquina ejecutando durante un tiempo prologado. Esta situación es en parte paliada gracias a disponer de aceleración gráfica que reduce considerablemente el tiempo requerido por los algoritmos.

De los recursos de hardware que hemos comentado anteriormente disponemos de un pc con un consumo de 33W, que utilizamos la mayor parte del tiempo para las tareas de desarrollo. En cuanto a los servidores tenemos un consumo de 500W.

Es un consumo muy superior, pero hemos de aclarar que los servidores tienen otros usuarios y que permanecen siempre funcionando, tengamos nosotros un proceso ejecutando o no. Por lo que el impacto de nuestro proyecto en su consumo es pequeño.

Suponiendo que hemos utilizado un 60 % del tiempo del proyecto nuestro pc y un 40 % esto nos deja un consumo medio de 220W durante las 580 horas del proyecto. Lo cual significa un consumo total de 127.6KWh o 38.28 Kg de CO2.

Un aspecto en el que nuestro proyecto, cumpliendo las dependencias con software de terceros, esta estructurado de forma que pueda ser reutilizado por otros que deseen utilizarlo y por lo tanto puede ser evitado el coste medioambiental de volver a desarrollarlo.

En lo que respecta a sostenibilidad ambiental valoraría este proyecto con un 7, ya que necesitamos una cantidad de energía para el desarrollo, aunque hemos intentando que el tiempo de ejecución fuera el menor posible para minimizar el gasto. Pero objetivo del proyecto no supone una diferencia en el impacto medioambiental de sus usuarios.

Capítulo 3

Adecuación a la especialidad de computación

En la especialidad de computación encontramos englobadas distintas disciplinas, como por ejemplo los gráficos, la algoritmia o los lenguajes de programación. En la realización de este trabajo nos centraremos en la inteligencia artificial y su aplicación.

Aun así la inteligencia artificial continua siendo un campo demasiado amplio para el contexto de este proyecto. Dentro de ella nos centraremos en distintos algoritmos supervisados de aprendizaje automático y en su uso para realizar tareas de clasificación sobre lenguaje natural y traducción.

Para ello, a partir de textos crearemos modelos que permitan representar las características de las palabras que forman nuestros textos de forma que sean relevantes para los distintos algoritmos.

A su vez, para disponer de datos con los que entrenar los algoritmos éstos han de gran tamaño. Los resultados obtenidos con nuestros modelos pueden ser incorporados a un sistema de traducción online, por lo que es importante que estas tareas se realicen eficientemente en tiempo para no hacer esperar a los usuarios, y consumiendo pocos recursos para poder servir el mayor número de peticiones posible. Para lograrlo utilizaremos técnicas aprendidas durante la especialidad en el ámbito de la algoritmia .

También tras examinar distintos lenguajes de programación seleccionamos python para realizar las tareas anteriores, el cual al ser un lenguaje multiparadigma nos permite aplicar conceptos de los lenguajes de programación, tales como scripts o uso de lambdas, con el objetivo de realizar código eficiente y mejorar su legibilidad.

Además, una parte significativa del proyecto es el diseño y la realización de distintos experimentos para determinar de manera objetiva cuales de las técnicas propuestas de ajusta mejor a las tareas a realizar. Este aspecto es tratado en distintas asignaturas de la especialidad, como puede ser Inteligencia Artificial, entre otras.

En resumen, los aspectos importantes de nuestro proyecto son la aplicación de técnicas de aprendizaje automático, la creación de eficiente de modelos para entrenar eficazmente estos algoritmos y el diseño de experimentos que nos permitan evaluar los resultados obtenidos.

Conocimientos de asignaturas utilizados

En las diferentes asignaturas de la especialidad hemos tratado conceptos que aplicaremos en la realización de este proyecto.

En primer lugar destacamos la asignatura de Aprendizaje automático, en la cual hemos visto si no todos los algoritmos utilizados en este proyecto si las bases teóricas en las que se fundamentan.

También han tenido gran importancia los conceptos vistos en la asignatura de Inteligencia Artificial sobre diseño de experimentos y como documentar y analizar los resultados obtenidos para poder seleccionar los modelos más adecuados para nuestro desarrollo y como ajustarlos.

Otros conceptos que hemos utilizado durante todo el desarrollo son las técnicas aprendidas en Algoritmia y Ampliación de Algoritmia para generar algoritmos eficientes tanto en tiempo como en memoria que nos permitan realizar las tareas de nuestro proyecto con los recursos disponibles.

Además los paradigmas estudiados en la asignatura de Lenguajes de programación nos permiten utilizar aspectos de los lenguajes de programación empleados que mejoran la calidad final del código obtenido.

En resumen, los conocimientos adquiridos en la especialidad de computación se ven reflejados en este proyecto en las técnicas de aprendizaje automático empleadas y en la forma de desarrollar el código buscando el resultado más eficiente y legible posible.

Competencias técnicas del proyecto

En la siguiente sección describimos las distintas competencias técnicas abordadas durante la realización de este proyecto y en que grado han sido tratadas.

- **CCO1.1**: Evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan llevar a cabo su resolución, y recomendar, desarrollar e implementar la que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

  Abordaremos esta competencia en profundidad, ya que crearemos modelos y algoritmos que nos permitan descomponer los textos que pretendemos traducir y una vez hemos acabado el proceso de clasificación nos permitan crear un texto final añadiendo el conocimiento aportado por los algoritmos. Es importante que estas tareas se realicen eficientemente ya que podemos trabajar con un gran volumen de datos y una mala implementación podría hacer imposible ejecutar los procesos con el hardware disponible.

- **CCO1.2**:Demostrar conocimiento de los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber aplicarlas para la creación, el diseño y el procesamiento de lenguajes

  Abordaremos un poco esta competencia. Pese a que no es nuestro objetivo explorar las posibilidades del lenguaje del programación utilizado, sino crear un sistema eficiente y funcional. Conocer y aplicar distintas herramientas presentes en el lenguaje nos permite mejorar la calidad del código creado, su legibilidad y su mantenibilidad.

- **CCO2.1**:Demostrar conocimiento de los fundamentos, de los paradigmas y de las técnicas propias de los sistemas inteligentes, y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen estas técnicas en cualquier ámbito de aplicación.

  Aplicaremos esta competencia en profundidad. El objetivo de este proyecto es la aplicación de técnicas de aprendizaje automático en el contexto de la traducción estadística y las etiquetas gramaticales.

- **CCO2.2**: Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano de una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente en los que están relacionados con aspectos de computación, percepción i actuación en ambientes o entornos inteligentes.

  Abordaremos un poco esta competencia. En el inicio de nuestro proyecto realizamos una fase de investigación para determinar cuales son las características principales de los lenguajes a tratar y cuales son las principales diferencias entre ellos que dificultan la tarea de traducción.

- **CCO2.4**: Demostrar conocimiento y desarrollar técnicas de aprendizaje computacional; diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las que se dedican a la extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

  Abordaremos esta competencia en profundidad. Durante la realización del proyecto experimentaremos con distintos algoritmos de aprendizaje automático con el objetivo de encontrar un modelo que se ajuste a las características del problema de completar etiquetas gramaticales simplificadas.

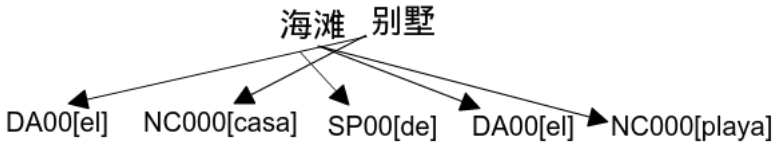
Capítulo 4

Ejemplo

En esta sección presentaremos un ejemplo del proceso que realiza nuestro sistema a partir de un fragmento de texto en chino para generar su correspondiente traducción al castellano.

En primer lugar utilizamos la herramienta *Moses* para generar una traducción simplificada de las palabras del fragmento de texto. Cada palabra de esta traducción está formada por una etiqueta la cual no dispone de información morfológica y del lema, o palabra base a partir de la cual podemos general la flexión correspondiente. En la siguiente figura podemos ver una representación de este proceso:

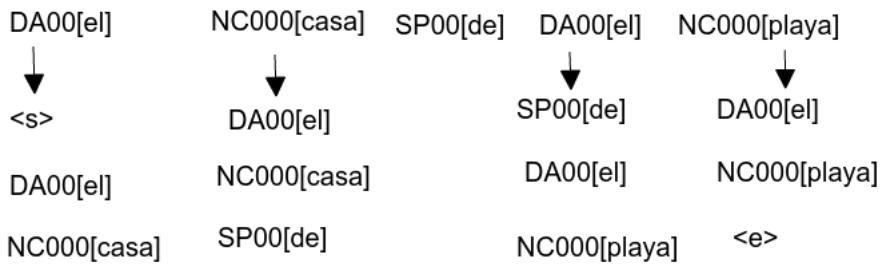
Figura 4.1: Ejemplo de traducción chino-castellano simplificado



Una vez hemos conseguido una traducción simplificada de nuestros datos procedemos a preprocesar los datos para poder realizar la clasificación.

Para ello separamos el fragmento de texto en ventanas de texto de texto de tamaño fijo, como vemos en la figura 4.2:

Figura 4.2: Creación de las ventanas de texto



Destacamos que para aquellas palabras que no pertenecen a una categoría que o tenga información morfológica en su etiqueta no generamos su ventana. En el caso del ejemplo no generamos la ventada de la palabra de.

Para poder utilizar estos datos en nuestros clasificadores necesitamos convertir nuestras ventanas en características numéricas. Para ello asignamos a cada palabra su índice en el vocabulario creado en la fase de entrenamiento.

Figura 4.3: Conversión de las ventanas en vectores numéricos

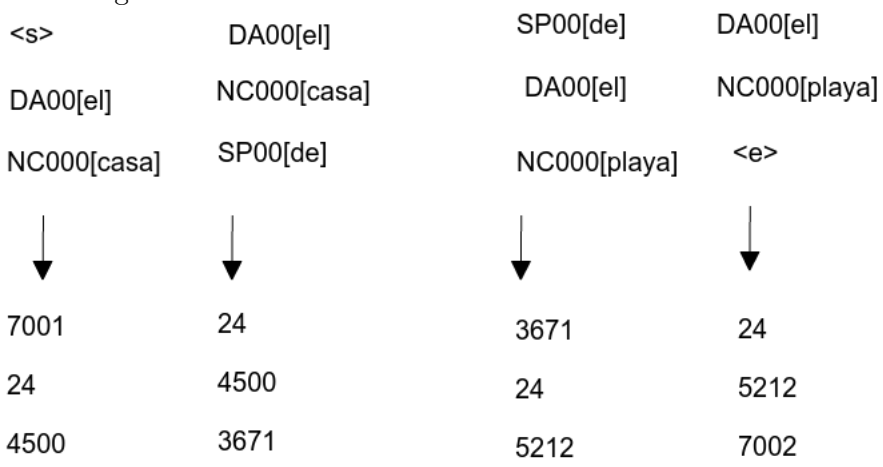


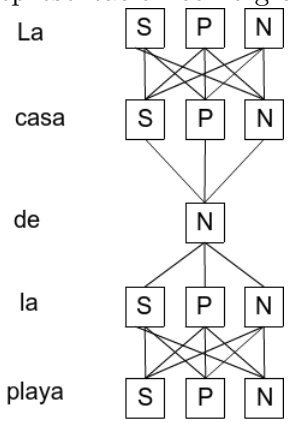
Tabla 4.1: Probabilidad de pertenecer a las distintas clases.

	Número			Género		
	S	P	N	M	F	N
DA00[el]	0,9	0,03	0,17	0,2	0,7	0,1
NC000[casa]	0,99	0,005	0,005	0,03	0,95	0,02
DA00[el]	0,76	0,24	0,1	0,003	0,997	0,002
NC000[]	0,6	0,38	0,02	0,4	0,55	0,05

Usamos los vectores creados con nuestros modelos y obtenemos su probabilidad de pertenecer a cada clase.

A partir de los resultados obtenidos del modelo sobre información de número generamos el grafo de relaciones entre las distintas posibilidades de cada palabra. La figura 4 muestra el grafo resultante:

Figura 4.4: Representación como grafo de la frase



Con nuestro grafo definido podemos calcular los k mejores caminos que representan las etiquetas de número con mayor probabilidad para nuestra frase según los resultados del clasificador.

Para cada uno de los k caminos creamos las etiquetas completas añadiendo la información de género. Usando los diccionarios de *Freeling* generamos las flexiones de las palabras.

En este punto para cada linea del texto original a traducir hemos generado k lineas, cada una con su correspondiente probabilidad. Para encontrar la mejor frase de las que hemos generado, utilizando la herramienta de *rescoring* de *Moses* combinamos las probabilidades que hemos obtenido anteriormente con el peso de cada posibilidad en un modelo de lenguaje.

Tras este proceso obtenemos finalmente la traducción a castellano de la entrada original en chino a nuestro sistema.

Capítulo 5

Base teórica

En esta sección explicaremos los conceptos teóricos tras las procesos utilizados en este proyecto, explicando brevemente su funcionamiento y su utilidad en la tarea a realizar.

5.1. Procesado del lenguaje natural

A lo largo de este documento utilizamos distintos conceptos de procesado del lenguaje natural para explicar nuestras aproximaciones a la tarea a realizar. En esta sección pretendemos definir estos conceptos para proporcionar una visión general sobre ellos.

N-gramas

Un n-grama es una secuencia de n elementos presente en una secuencia dada. Por para cualquier fragmento de texto dado, cada palabra que contiene es un n-grama de tamaño 1 o unigrama.

A partir de n-gramas podemos desarrollar modelos probabilísticos llamados modelos de n-gramas. Estos modelos se basan en asumir que nuestro sistema es un sistema estocástico, es decir que no es un sistema impredecible debido a una variable aleatoria. Con esta asunción podemos aplicar la propiedad de Markov a nuestros datos.

La propiedad de Markov define que la probabilidad de un elemento solo depende de un limitado historial previo, o generalizando, que la probabilidad de un elemento solo depende de los n elementos previos.
De esta forma podemos calcular la probabilidad de las palabras de nuestro corpus de las formas siguientes:

- En el caso de un modelo de unigramas podemos calcular su probabilidad utilizando la siguiente expresión:

$$P(w) = \frac{x}{t}$$

Donde x es el número de repeticiones de la palabra w en el texto y t es el número de palabras en el texto.

- En el resto de casos, asumiendo la independencia de las palabras de nuestro corpus, podemos calcular su probabilidad como el producto de sus propiedades condicionadas, de la siguiente forma:

$$P(w_1, w_2, ..., w_n) = P(w_1|w_2, ..., w_n) \cdot P(w_2|w_3 \cdot ... \cdot w_n)...P(w_{n-1}|w_n) \cdot P(w_n)$$

O escrito de otra forma:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k_1}^{k-1})$$

Dependiendo de la n que escojamos para nuestro modelo, éste presentará características diferentes:

- A medida que aumentamos n la precisión del modelo aumenta, al tratarse de secuencias más especificas y tener más información del contexto.

- A mayor n menor es la frecuencia de los n-gramas que recogemos, ya que debe repetirse una secuencia de longitud mayor.

Es importante utilizar un corpus que refleje las características del dominio sobre el que vamos a trabajar. Un corpus demasiado especifico no reflejará correctamente el idioma, mientras que un corpus demasiado general no permitirá representar un dominio especifico.

Sintagmas

Un sintagma es un conjunto de palabras que desarrollan una función dentro de la oración, por ejemplo en la oración el *el artículo ha sido publicado* el sintagma *el artículo* ejerce la función de complemento directo en la oración.

Todo sintagma tiene un núcleo que define las funciones que puede llevar a cabo. El resto de palabras funcionan como complementos de ese núcleo. Y la propiedad realmente importante para la tarea que abordamos en nuestro proyecto es que la información morfológica de las palabras del sintagma es compartida y viene definida por la información que pretenda expresar el núcleo.

Tomemos como ejemplo la oración *El pequeño niño compra caramelos* y concretamente el sintagma nominal *El pequeño niño*. Si variamos la información morfológica que alguna de las palabras del sintagma sin variar las demás deja de tener sentido no sería correcto utilizar expresiones como *las pequeño niño* ni *el pequeña niño* ni *el pequeño niña*

Etiquetado gramatical y lemas

El etiquetado gramatical consiste en métodos para representar la información de una palabra por medio de una etiqueta. Diversas informaciones pueden ser representadas de esta forma, desde el tipo de palabra, hasta información más detallada como puede ser el tiempo y la persona en el caso de los verbos o si se trata de un nombre común o propio.

A su vez para representar completamente la información de una palabra dada necesitamos conocer su raiz, aquella forma a partir de la cual añadiendo afijos y aplicando reglas de conjugación podemos recuperar la palabra original. Esta raíz de cada palabra recibe el nombre de lema.

La importancia de estos conceptos en nuestro desarrollo es el hecho de permitirnos simplificar la información de las palabras de forma no ambigua y posteriormente recuperar esa información. Imaginemos el caso de los determinantes *el, la, lo, los, las* que podemos etiquetar como *DAMS00[el], DAFS00[el], DANS00[el] ,DAMP00[el], DAFP00[el]* respectivamente, donde las posiciones 3 y 4 de la etiqueta indican la información de género y número y *el* es el lema de todas ellas. Si eliminamos la información de género y número de nuestras etiquetas obtenemos *DA00[el], DA00[el], DA00[el] ,DA00[el], DA00[el]*. Es decir, las cinco palabras anteriores pasan a ser representadas con un único par de etiqueta y lema. El hecho de reducir la información presente en las palabras del texto permite hacer la tarea de traducción más simple al no tener tener que representar los modelos esta información.
Durante el desarrollo de este proyecto hemos utilizado el analizador sintáctico *Freeling*, el cual utiliza el formato de etiquetas *EAGLES*. En la tabla 5.1 mostramos los diferentes tipos de palabras representados junto con la posición en la que se representa la información morfológica en la etiqueta:

Tabla 5.1: Tipos de palabras representados y la posición de su información morfológica

Tipo	Posición Género	Posición Número	Ejemplo
Adjetivos	3	4	AQ0CS0[alegre]
Conjunciones	-	-	CC[y]
Determinantes	3	4	DA0MS0[el]
Nombres	2	3	NCMS000[perro]
Pronombres	3	4	PP3FS000[él]
Adverbios	-	-	RG[hábilmente]
Preposiciones	-	-	SP[ante]
Verbos	6	5	VMII3S0[hablar]
Número	-	-	Z[1]
Interjección	-	-	I[ay]
Puntuación	-	-	Fp[.]

5.2. Sistemas de traducción basados en frases

Los sistemas de traducción nos permiten a partir de un texto en un idioma y su correspondiente traducción entrenar un modelo de traducción que nos permita traducir cualquier texto de este idioma al segundo idioma, el idioma objetivo. Para ello se basan en el cálculo de la probabilidad condicionada de las palabras del idioma objetivo dada una palabra o conjunto de palabras del lenguaje original.

Este proceso puede ser realizado a nivel de palabras individuales en el texto, como es el caso de los Modelos *IBM* o a nivel de frase[Koehn et al., 2003] en el que utilizamos para la traducción grupos de palabras a las que asignamos una probabilidad como si se trataran de un solo elemento. Este enfoque presenta diversas ventajas como es que es mucho más sencillo capturar expresiones del idioma ya que se tratan como una única entidad y se les asigna una única probabilidad.

La probabilidad asignada a cada palabra o conjunto de palabras es calculada a partir de la frecuencia en la que éstas aparecen en el texto. Para ello creamos una tabla de frases en la que se recogen por cada par de palabra y traducción su probabilidad. Esta tabla generalmente será mayor al texto con el que se crea ya que ha de contener todas las posibilidades aparecidas entre palabras individuales en el texto hasta el tamaño máximo de frase determinado en el sistema. Para realizar esta tarea es necesario entrenar un sistema de alineamiento que nos permita relacionar una palabra con su correspondiente traducción. La dificultad de esta tarea aumenta al tratar idiomas que no tengan una base común ya que pueden ser muy diferentes en la organización de las frases. En la realización del proyecto hemos utilizado como sistema de referencia *Moses*[Koehn et al., 2007] utilizando como alineamiento *GIZA++* y como optimización en el entrenamiento *Minimun Error Rate Training(MERT)*.

5.3. Embedding

La tarea de entrenar algoritmos de aprendizaje automático a partir de texto es compleja, en parte debido a las características de los datos. Representaciones basadas en un vocabulario de apariciones en el texto proporcionan valores discretos y en un número reducido de dimensiones.

Para el problema de dimensionalidad existen formas de expresar las palabras de nuestro corpus como un conjunto mayor de características, siendo una de las técnicas más utilizadas el denominado *one hot encoding*.

Este método consiste en para cada palabra crear una lista de tamaño igual al vocabulario, en que cada posición de la lista contiene un 0 excepto la posición igual al índice de la palabra en el vocabulario.

Por ejemplo, si tenemos un vocabulario de 5 palabras en el cual la palabra *el* tiene indice 2 su presentación usando este sistema sería $[0, 0, 1, 0, 0]$.

Usando lo anterior tenemos un mayor número de dimensiones, pero la representación continua siendo en valores discretos y no tenemos más información de la palabras que su posición en el diccionario.

Para resolver esta situaciones recurrimos al uso de *embeddings*, los cuales nos permiten representar cada palabra en un espacio continuo de mayor dimensionalidad.
Esta técnica consiste en un algoritmo de aprendizaje no supervisado, en el cual realizaremos la siguiente operación para cada palabra de nuestro texto:

$$E_i = W * P_i$$

Donde :

- E_i es la representación de la palabra en el *embedding* de la iésima palabra.
- W es el vector de parametros que pretendemos optimizar.
- P_i es la representación en *one hot encoding* de la iésima palabra.

De esta forma obtenemos por cada palabra a tratar definida como un punto en el espacio, sobre el que podemos aplicar el resto de algoritmos para hallar relaciones entre ellos.

5.4. Redes convolucionales

Utilizando el resultado de lo anterior podemos realizar clasificación pero observamos que los resultados no son óptimos, y que el conjunto de puntos con el que contamos no proporciona información suficiente. Sería necesario encontrar alguna forma de presentar la similitud entre distintas palabras. Ese es el objetivo de aplicar redes convolucionales a nuestra tarea.

Estas redes se basan en como el cerebro de los animales procesa la información visual aplicando operaciones a secciones del campo visual buscando relaciones entre lo que percibe.

De la misma forma en estas redes aplicamos *filtros* repetidamente sobre la representación de nuestros datos. Definimos *filtros* como la matriz de parámetros que buscamos optimizar y la cual aplicamos sobre nuestra entrada.

En primer lugar utilizando estos *filtros* definimos el concepto de Correlación como:

$$Z_i = \sum_i^S w_i X_{i'+i-1}$$

Donde:

- Z_i es el iésimo elemento del resultado de la correlación.
- X_i es el iésimo elemento de los datos de entrada
- S es el tamaño de la entrada
- w_i es el iésimo elemento de la matriz de parámetros.

La utilidad de este concepto para nuestra tarea es aplicado sobre una matriz que representa una secuencia de palabras nos proporciona una medida de la similitud entre ellas.

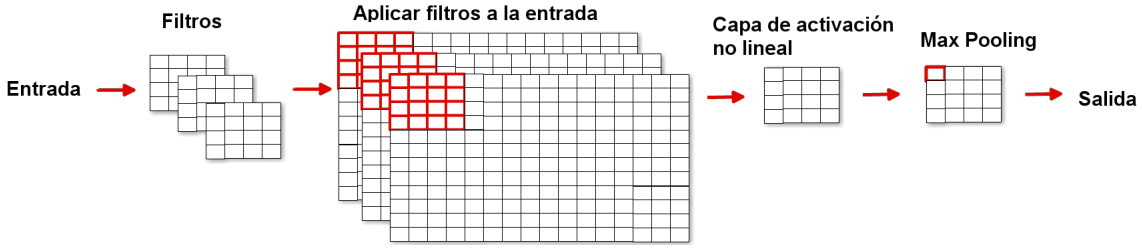
Definimos convolución como la correlación invirtiendo el orden en que aplicamos nuestros *filtros*, de forma que obtenemos:

$$Z_i = \sum_i^S X_{i'+i-1} w_{S-i+1}$$

Pero este concepto no es el único que se lleva a cabo en una red convolucional. En la figura mostramos una representación de este proceso:

- **Convolución:** En este paso aplicamos diversos *filtros* sobre nuestros datos. El número de *filtros* a utilizar y el desplazamiento entre *filtros* son hiperparamentros de la red.
- **Capa de activación no lineal:** En esta capa aplicamos un paso con una capa utilizando una función de activación no lineal. Un ejemplo de función comúnmente utilizado es relu, la cual podemos definir como:

Figura 5.1: Ejemplo del proceso dentro de una red convolucional



$$f(x) = argmax(x, 0)$$

- **Max Pooling:** Tras aplicar los pasos anteriores hemos obtenido una matriz por cada uno de los *filtros* aplicados. En este paso escogemos para cada uno de los valores de la matriz de salida el valor máximo de entre todos los obtenidos para esa posición tras los *filtros*. Este es el método más común, pero existen otras alternativas, como utilizar la media de estos valores.

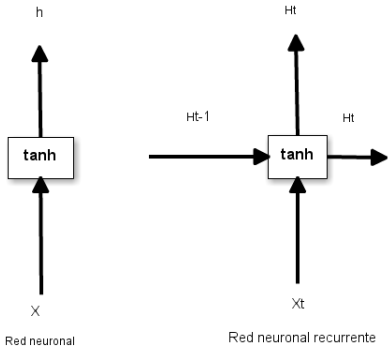
De esta forma nuestra red, aplicando la convolución, nos permite a partir de unos datos representados como una matriz obtener un matriz de dimensionalidad menor pero que contiene información sobre la similitud entre las palabras de la secuencia.

5.5. Long-short term memory networks (LSTM)

En algunos problemas los datos de los que disponemos no son características o atributos de un elemento sino un conjunto de sucesos secuenciales. En estos casos las redes neuronales clásicas no nos permiten guardar información sobre el contexto de los datos y utilizarlo para los siguientes. En los campos en que esta situación es común como el tratamiento de texto son comúnmente empleadas las redes neuronales recurrentes.

En estas redes cada nodo no funciona como un *multilayer perceptron* convencional que recibe datos de la capa de entrada y produce una salida, sino que además de la entrada recibe el resultado del nodo anterior y lo tiene en cuenta para generar su resultado. Finalmente este resultado es enviado al siguiente nodo de la capa. Podemos ver en este ejemplo un nodo de una red recurrente simple:

Figura 5.2: Representación de un nodo de la capa oculta de un MLP y de una red recurrente



En ambos casos se utiliza la misma función de activación, *tanh*, pero en el caso de la red recurrente la información de los otros datos introducidos es incluida en el cálculo.Definimos la función *tanh* o tangente hiperbólica como:

$$sinh(x) = \frac{e^x - e^{-x}}{2}$$

$$cosh(x) = \frac{e^x + e^{-x}}{2}$$

$$tanh(x) = \frac{sinh(x)}{cosh(x)}$$

En la tarea que nos ocupa en este proyecto esto de gran utilidad ya que el hecho determinar la información morfológica de una palabra depende en gran medida de las palabras que la rodean, por ejemplo en el caso de los verbos donde se debe asegurar la concordancia con el sujeto.

Sin embargo, este modelo de red presenta una característica que no la hace apropiada para muchas situaciones y es que al utilizar únicamente información del nodo inmediatamente anterior, la información de una palabra puede no propagarse a suficientes elementos de la secuencia.

Imaginemos en el caso que tratamos, que recibimos la linea *El gorro de ducha.* Al tratar la palabra *ducha* recibimos información sobre *de* que es neutra en número, cuando el elemento de la secuencia que nos aporta más información es *gorro*

En casos en que necesitamos retener la información durante más tiempo es conveniente utilizar *Long short-term memory (LTSM)* [Hochreiter and Schmidhuber, 1997] las cuales nos permiten almacenar información sobre el contexto por más nodos. Como consecuencia de ello hemos de tener en cuenta que son necesarios mecanismos que permitan olvidar esa información cuando ya no sea relevante.

Todos los nodos de la capa oculta de la red comparten el denominado estado de la celda, el cual contiene información de todos los elementos previos en la secuencia, mientras la puerta de olvido (*forget gate*) lo permita.

Esta puerta consiste en una red neuronal con función de activación *sigmoide* que indica qué cantidad de información es admitida a partir de los estados anteriores y de la entrada. Su salida está comprendida entre 0 y 1, siendo 1 permitir toda la información y 0 olvidarla completamente. Un ejemplo de función *sigmoide* empleada podría ser la función logística, definida como:

$$P(x) = \frac{1}{1+e^{-t}}$$

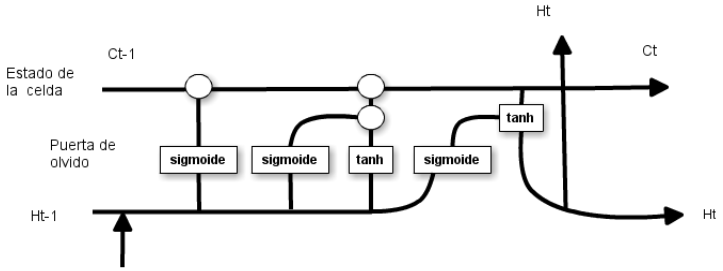
Esto resulta de gran utilidad en nuestra tarea, ya que permite que al encontrar cambios en la información como pueden ser signos de puntuación, el estado anterior se descarte para evitar información contradictoria.

Una vez escogida la información del estado se va a utilizar se actualiza el estado con la información aportada por la entrada del nodo. Para ello generalmente se combina una capa con activación *sigmoide* y otra capa *tanh* con el estado de la celda para generar su versión actualizada.

Con esto se ha actualizado el estado pero antes de retornarlo como salida hemos de filtrarlo, para lo que se utiliza otra capa con activación *sigmoide* que produce el filtrado y posteriormente una capa con activación *tanh* que tiene como salida valores comprendidos entre -1 y 1. Este valor es el que finalmente devolverá la red.
Podemos ver una representación esquemática de esta estructura:

Existen variantes de este método siendo una de las mas utilizas las *General Recurrent Units(GRU)* en la cuales tenemos un proceso similar donde el lugar de tener dos capas separadas para escoger la información del estado y actualizarla existe una única capa llamada puerta de actualización (*update gate*).

Figura 5.3: Ejemplo de nodo de una red *LSTM*



5.6. *Naive Bayes*

Uno de los algoritmos de aprendizaje automático más comunes es el de *Naive Bayes* el cual nos permite crear un modelo de clasificación sencillo basado en aprendizaje supervisado. Este algoritmo resulta de gran utilizan en ámbitos como el *big data* donde el gran volumen de datos hace imposible entrenar otro tipo de modelos en poco tiempo o por limitaciones de infraestructura técnica.
Para cada valor de nuestras características calculamos su probabilidad. Por ejemplo si pretendemos clasificar si lloverá por la tarde algunas características que podemos utilizar es la presencia o no de nubes actualmente en el cielo y si llovió el día anterior. Digamos que tenemos las muestras de la tabla:

Tabla 5.2: Ejemplo de observaciones

Observación	Nubes	Llovió el día anterior	Lluvia hoy
1	Sí	No	No
2	Sí	Sí	Sí
3	No	Sí	No
4	Sí	No	Sí

Lloverá hoy	Sí	No
Nubes	1	0
Lluvia ayer	0.5	0.5

No lloverá hoy	Sí	No
Nubes	0.5	0.5
Lluvia ayer	0.5	0.5

Asumimos que todas las probabilidades podemos realizar la clasificación de nuestros datos aplicando la regla de Bayes de la forma:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{PB}$$

Asumiendo la independencia de los datos y teniendo en cuenta que $P(B)$ no depende de las características de entrada podemos calcular la probabilidad de cada clase como:

$$P(A|b_1, b_2, ..., b_n) = P(A) \cdot P(b_1, b_2, ..., b_n) = P(A) \cdot \prod_{i=1}^{n} P(b_i|A)$$

Finalmente, la clase que queremos clasificar es aquella que tenga la probabilidad máxima, por lo que la fórmula resultante será:

$$argmax_{i=1}^{n}(P(c_i)cdot \prod_{j=1}^{m} P(a_j|c_i)$$

Volviendo al ejemplo anterior para un día en el que no haya nubes y el día anterior no lloviera obtenemos las probabilidades:

$$P(llover) = 0 \cdot 0{,}5 \cdot 0{,}5 = 0$$
$$P(nollover) = 1 \cdot 0{,}5 \cdot 0{,}5 = 0{,}25$$

Elegimos siempre la probabilidad mayor, por lo que en este caso la clasificación del algoritmos es que no lloverá.

5.7. *Support Vector Machines*

En esta sección presentamos otro algoritmo de aprendizaje automático basado en aprendizaje supervisado, los *support vector machines*

Representamos nuestros datos de entrada como puntos en un espacio. Para cada una de las clases que pretendemos clasificar tenemos dos opciones, pertenecer a esa clase o pertenecer a cualquiera de las demás. Nuestro objetivo es encontrar un hiperplano que separe esos puntos que pertenecen a la clase de aquellos que no. En la figura 1 podemos ver una representación:

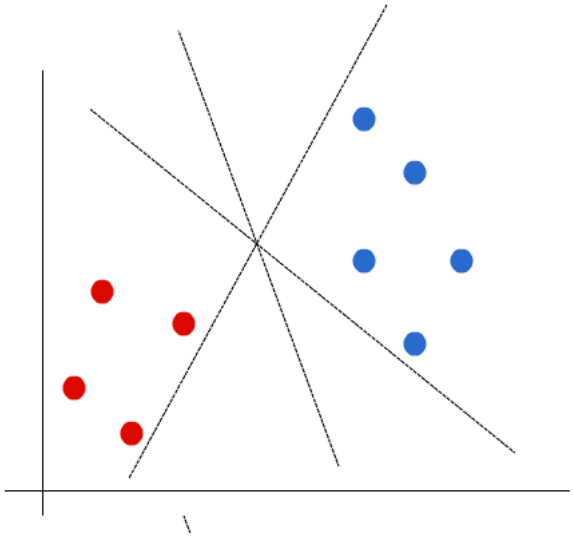
Como podemos ver en la figura existen distintos planos que nos permiten diferenciar nuestras clases. Por ello hemos de encontrar el hiperplano de margen máximo.

Definimos este hiperplano como el hiperplano que maximice la expresión:

$$d_+ + d_-$$

Donde d_+ es la distancia mínima a un punto del espacio que pertenece a la clase y d_- es la distancia mínima a un punto que no pertenece a ella

Figura 5.4: Hiperplanos entre las diferentes clases



Es decir, buscamos el plano que se encuentre a la mayor distancia de los puntos más cercanos a él. Estos puntos más cercanos reciben el nombre de vectores soporte o *support vector machines en inglés*.

En la figura podemos ver un ejemplo de hiperplano de margen máximo con sus márgenes:

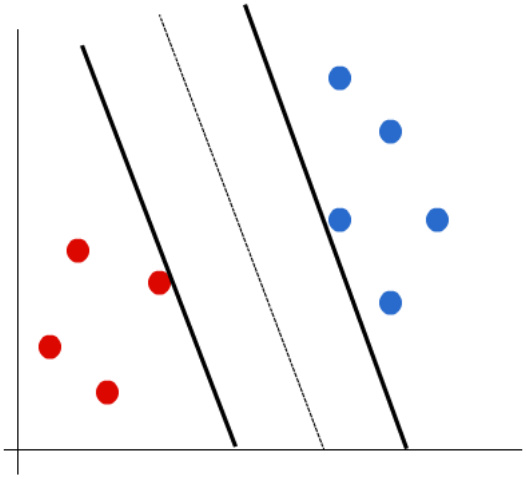
Una vez definido un plano podemos calcular un vector w normal a ese plano. Con él para un punto cualquiera x en el espacio calculando su proyección sobre w utilizando la expresión:

$$w \cdot x$$

Observando en que semiespacio se encuentra la proyección podemos predecir a que clase pertenece ese punto.

El método anterior nos permite crear un clasificador siempre que podamos separar nuestras clases de forma lineal, pero esto no siempre sucede. La figura 3 muestra un ejemplo en el que no es posible realizar una clasificación adecuada:

Figura 5.5: Hiperplanos de margen máximo



Como hemos visto anteriormente la forma que clasificar a qué clase pertenece un punto es calculando su proyección sobre un vector perpendicular al plano. En los casos en los que esto no da los resultados esperados ¿por qué no calculamos esta proyección en un nuevo espacio?

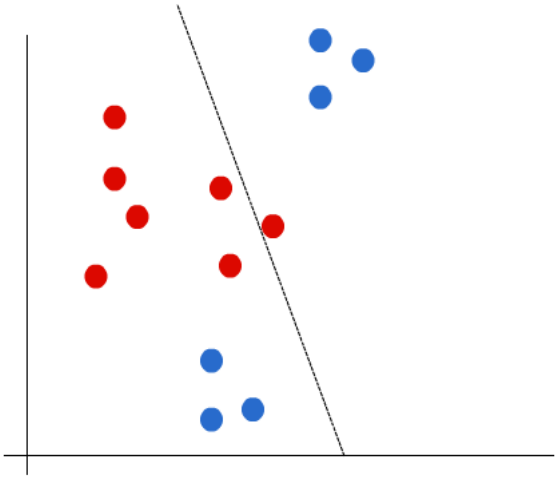
Definimos una nueva función K tal que:

$$K(w, x) = \phi(w) \cdot \phi(x)$$

De forma que $K(w, x)$ continua siendo la proyección del punto sobre el vector normal al plano pero en el espacio definido por la función ϕ. A esta función K la llamaremos función de kernel.

Si aplicamos nuestra función K con ϕ igual a la identidad, llamada kernel lineal, obtenemos el clasificador presentado anteriormente.

Figura 5.6: Datos no separables linealmente



Otros ejemplos de funciones de kernel utilizadas son:

- **kernel polinómico:** En la que K se define como el siguiente polinomio:

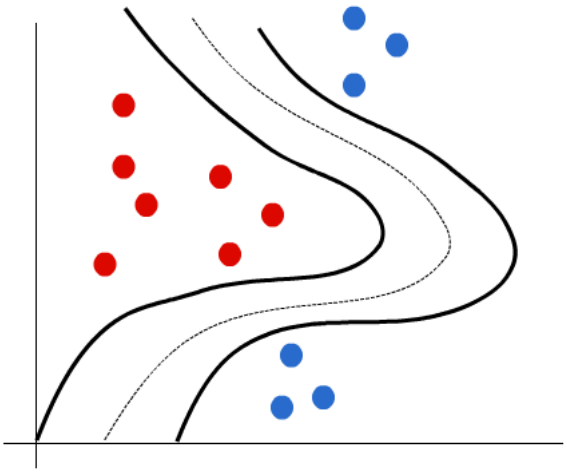
$$K(w, x) = [1 + w \cdot x]^k$$

- **Función de base radial:** En la cual medimos la distancia del punto. Todos los puntos a la misma distancia obtienen el mismo valor de esta función. La definimos como:

$$K(w, x) = e^{[\frac{1}{2}||w-x||^2}$$

En la figura 4 podemos ver como usando un kernel polinómico podemos clasificar de forma mucho más ajustada los datos anteriores:

Figura 5.7: Ejemplo de kernel polinómico



5.8. Métodos de ensemble y Random Forest

En las secciones anteriores hemos presentado distintos algoritmos de aprendizaje automático basados en crear un modelo que se ajustara a nuestros datos. En esta sección vamos a abordar una familia de algoritmos que tiene un enfoque diferente para esta tarea, los métodos de *ensemble*, y dentro de ellos el método utilizado durante el proyecto, *random forest*.

Los métodos de *ensemble* consisten en entrenar un conjunto de modelos sobre nuestros datos para finalmente combinar sus resultados para obtener el resultado final. Generalmente en tareas de clasificación la respuesta mayoritaria de los modelos es la elegida mientras que para tareas de regresión suele utilizarse la media de éstas.

Un problema al que nos enfrentamos al entrenar estos métodos es como repartimos los datos de los que disponemos entre los distintos modelos. Necesitamos que cada uno se entrene con información distinta para poder generar modelos distintos y que aporten información nueva sobre el problema.

Para abordar este problema existen distintos métodos de agregar diversidad adicional a nuestros datos de forma que cada modelo resultante sea único. Los más utilizados son:

- ***Bagging:*** Esta técnica consiste en crear nuevos subconjuntos de datos a base de añadir repeticiones de los mismos.Esto nos permite reducir la varianza conjunta y es indicado para modelos individuales que tiendan al sobreajuste, como por ejemplo las redes neuronales.

- ***Boosting:*** Esta técnica consiste en entrenar cada uno de los modelos individuales con todo el conjunto de datos del que disponemos pero dando una ponderación mayor a aquellas observaciones que han sido clasificadas incorrectamente por modelos anteriores. Nos permite reducir el *bias* conjunto a partir de modelos individuales que tengan mucho **bias** y poca varianza.

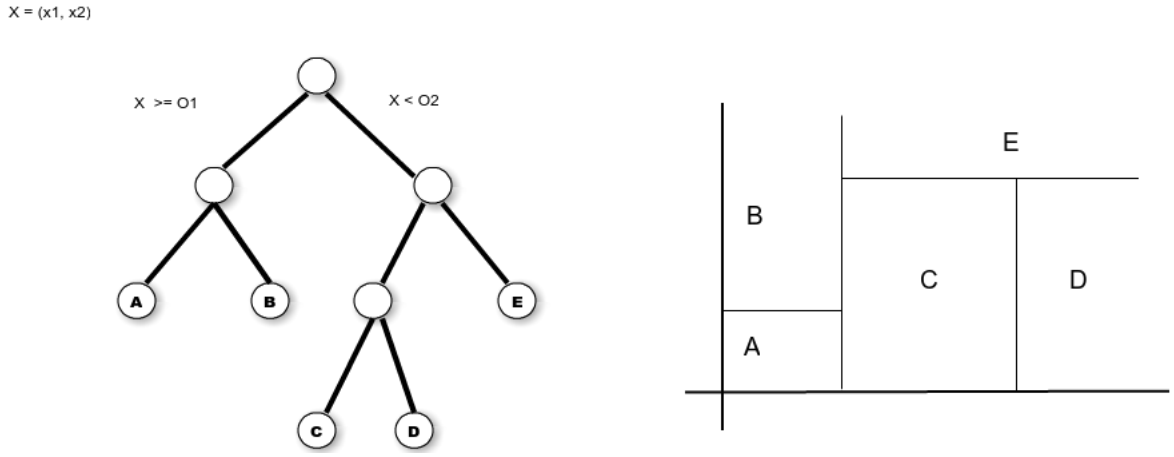
Uno de los más utilizados dentro de los métodos de *ensemble* son los *random forest*, en los cuales cada modelo individual es un árbol de decisión.

Un árbol de decisión es una estructura de datos formada por nodos, los cuales pueden tener otros nodos relacionados los cuales reciben su salida(hijos) o no tener ningún otro nodo dependiente en cuyo caso llamaremos hoja.

Para todo nodo de la estructura realizamos una comprobación sobre los datos de entrada Del resultado de esta comprobación depende a cual de los nodos dependientes del nodo actual enviaremos los datos.

Cuando durante la ejecución lleguemos hasta un nodo sin hijo entonces interpretaremos el resultado como la predicción de a que clase pertenece la información recibida como entrada.

Figura 5.8: Ejemplo de árbol de decisión



Como vemos en la imagen en la figura 1 hemos dividido el espacio en el que se sitúan nuestros datos en las distintas hojas de nuestro árbol de forma que para cualquier elemento que recorra nuestro árbol podemos predecir a que región del espacio pertenece a partir de la hoja en la que acabó su recorrido.

El proceso básico en **random forest** es para cada árbol de nuestro sistema generar mediante *bagging* un subconjunto de datos de tamaño n.

Generamos un subárbol escogiendo recursivamente para cada nodo m variables de nuestros datos aleatoriamente y escogemos de ellas el punto que mejor separa nuestros datos. Repetimos este proceso hasta conseguir una altura máxima del árbol o un número mínimo de elementos a clasificar.

Finalmente combinamos la salida de todos los árboles para generar el resultado final.

Capítulo 6

Análisis de los datos

En esta sección presentaremos los aspectos más relevantes del corpus de texto utilizado durante la realización de los experimentos.

Para asegurar la validez de los resultados obtenidos hemos utilizado el mismo corpus que se utilizó en el articulo original en el que basa este proyecto [Costa-Jussà, 2015]. Este corpus consiste en fragmentos extraídos de discursos de la ONU[Rafalovitch et al., 2009]. Para cada uno de los fragmentos disponemos de sus correspondientes traducciones en chino y castellano.

Tabla 6.1: Tamaño de los conjuntos del corpus

	Lineas	Palabras
Entrenamiento	58.688	2.297.656
Desarrollo	990	43.489
Validación	1.010	44.306

Como podemos ver en la tabla 1 disponemos de alrededor de 60.000 lineas de texto separados en tres conjuntos de la siguiente forma:

- **conjunto de entrenamiento:** El conjunto de mayor tamaño, lo utilizamos para ajustar los distintos algoritmos de clasificación con los que experimentamos.

- **conjunto de desarrollo:** Este conjunto es el utilizado en todas aquellas tareas que requieran una fuente de datos adicional al entrenamiento.

- **conjunto de validación:** Para medir los resultados obtenidos utilizamos este tercer conjunto el cual no es utilizado durante el proceso de entrenamiento.

En la tabla 1 hemos presentado el número total de palabras que forman nuestro corpus pero no todas ellas tienen flexiones en función de la información morfológica. Un ejemplo de ello son las preposiciones como la palabra *de*.

En primer lugar hemos de decidir sobre que tipos de palabras realizaremos nuestros experimentos. Consultando la documentación de la herramienta *Freeling* utilizada para etiquetar las palabras en castellano vemos que las categorías que sí presentan flexiones en función de la morfología son los determinantes, los nombres, los verbos, los pronombres y los adjetivos.

Con los tipos de palabras definidos podemos analizar las datos de los que disponemos. En primer lugar analizamos la información de número en nuestros datos:

Tabla 6.2: Distribución de las clases de número en el conjunto de entrenamiento

	Total	Singular(%)	Plural(%)	Invariable(%)
determinantes	340.739	61,80	38,19	0,01
nombres	57.1053	67,75	31,92	0,33
verbos	219.638	41,04	28,46	30,50
pronombres	43.806	12,21	8,09	79,70
adjetivos	185.107	63,38	36,38	0,24

Tabla 6.3: Distribución de las clases de número en el conjunto de desarrollo

	Total	Singular(%)	Plural(%)	Invariable(%)
determinantes	6.534	61,75	38,25	0
nombres	11.025	66,93	32,68	0,39
verbos	5.630	42,38	27,06	30,56
pronombres	1.079	9,55	7,04	83,41
adjetivos	3.129	60,95	38,69	0,36

Tabla 6.4: Distribución de las clases de número en el conjunto de validación

	Total	Singular(%)	Plural(%)	Invariable(%)
determinantes	6.858	60,95	39,05	0
nombres	11.347	65,51	34,14	0,35
verbos	4.629	40,53	29,66	29,81
pronombres	1.015	9,66	9,16	81,18
adjetivos	3.375	58,34	41,45	0,21

Como podemos ver en las tablas la distribución de las clases en los tres conjuntos es muy similar. También podemos observar que el porcentaje de palabras singulares suele ser el mayoritario excepto en el caso de los pronombres en los que predominan los neutros.

También podemos observar que para determinantes, nombres y adjetivos casi no hay presencia de *invariables*. Este hecho se puede producir por ser palabras que se encuentran en el sujeto de la oración e indican quién hace la acción por lo que representan más explícitamente esta información

En el caso del género observamos los siguientes resultados:

Tabla 6.5: Distribución de las clases de género en el conjunto de entrenamiento

	Total	Femenino(%)	Masculino(%)	Invariable(%)
determinantes	340.739	53,24	38,36	8,40
nombres	571.053	52,19	46,52	1,29
verbos	219.638	14,24	12,75	73,01
pronombres	43.806	4,28	6,91	88,81
adjetivos	185.107	21,69	24,14	54,17

Tabla 6.6: Distribución de las clases de género en el conjunto de desarrollo

	Total	Femenino(%)	Masculino(%)	Invariable(%)
determinantes	6.534	51,18	41,20	7,62
nombres	11.025	51,39	46,78	1,83
verbos	5.630	11,10	13,20	75,70
pronombres	1.079	3,89	6,12	89,99
adjetivos	3.129	23,19	23,54	53,27

Tabla 6.7: Distribución de las clases de género en el conjunto de validación

	Total	Femenino(%)	Masculino(%)	Invariable(%)
determinantes	6.858	52,07	40,76	7,17
nombres	11.347	51,72	46,56	1,72
verbos	4.629	12,18	13,23	74,59
pronombres	1.015	4,24	7,39	88,37
adjetivos	3.375	24,68	24,21	51,11

En las tablas vemos que como en el caso de la información de número la distribución de las clases entre los conjuntos es bastante similar aunque vemos que se reparten de forma distinta en este caso.

Observamos como la clase mayoritaria para verbos, pronombres y adjetivos es *invariables* mientras que para determinantes y nombres la representación de esta clase es muy baja.

Para estas dos categorías de palabra en cambio vemos que aunque el porcentaje de *masculinos* es mayor está bastante igualada la presencia de ambos géneros en los datos.

El hecho de que tengamos mayoritariamente palabras Invariantes es un indicador de como de dependientes del contexto son las palabras de esa categoría y de como de acertados serán las clasificaciones generadas por nuestro sistema.

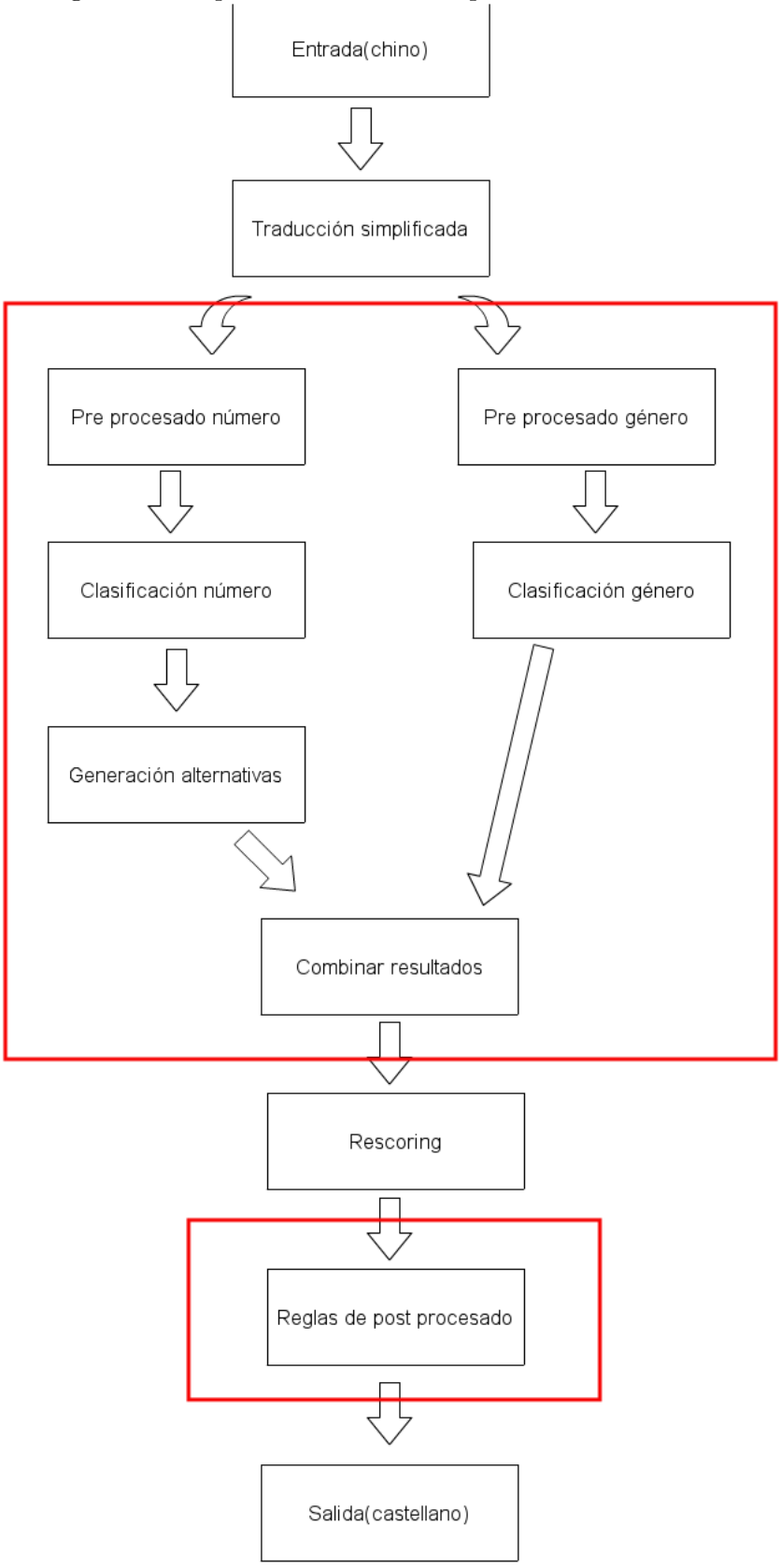
Capítulo 7

Arquitectura

En este capitulo describimos el sistema definitivo creado para la tarea de este proyecto, explicando que sucede en cada uno de sus pasos. En la figura 8.1 podemos ver una representación de la relaciones entre los diferentes procesos desde la entrada al sistema en chino hasta la salida de en castellano.

Haremos especial hincapié en las fases del proceso desarrolladas específicamente para este proyecto, marcadas en la figura 7.1 en rojo. Las fases no marcadas son aquellas en las que utilizamos *Moses*, ya que las tareas de traducción quedan fuera del alcance de nuestro proyecto.

Figura 7.1: Representación de la arquitectura desarrollada



7.1. Sistema de traducción

La primera fase a realizar es entrenar un modelo de traducción que nos sirva como base sobre la que aplicar la clasificación y de referencia para medir la mejora de la traducción tras aplicar nuestro sistema. Para ello utilizaremos ***Moses***[Koehn et al., 2007], la cual nos permite entrenar y optimizar un modelo de traducción a partir de un corpus de texto paralelo.

En primer lugar entrenaremos un modelo de traducción chino a castellano sin ningún tipo de modificación en el texto. El resultado obtenido por este modelo lo tomaremos como referencia para nuestros resultados.
En segundo lugar, utilizando el analizador de lenguaje ***Freeling*** [Padró and Stanilovsky, 2012, Padró, 2011, Padró et al., 2010, Atserias et al., 2006, Carreras et al., 2004] realizamos las siguientes representaciones del texto, que utilizaremos para entrenar distintos modelos:

- **Texto original:** Es el texto sin modificaciones, en el que cada palabra en chino se corresponde con su correspondiente traducción en castellano. El modelo creado con esta representación nos servirá como base para medir los resultados generados.

- **Texto simplificado:** En esta representación eliminamos de las etiquetas toda la información referente a la morfología de la palabra, género y número. El modelo obtenido nos permitirá acotar la mejora máxima que puede proporcionar nuestro sistema, ya que al eliminar esta información no estamos creando errores a causa de ellas y sería equivalente a predecirla correctamente en todos los casos

- **Género simplificado:** En esta representación eliminamos la información sobre género del texto en castellano. Utilizaremos este modelo como cota de la mejora posible de nuestro sistema aplicado únicamente al género.

- **Número simplificado:** De forma análoga al género simplificado, este modelo nos permite conocer la mejora máxima que podemos obtener utilizando nuestro sistema únicamente sobre número.

Tabla 7.1: Ejemplo de representaciones de texto utilizadas

Modelo	**Texto**
Original	La casa de la playa
Texto etiquetado	DA0FS0[el] NCFS000[casa] SPS00[de] DA0FS00[el] NCFS000[playa]
Simplificado	DA00[el] NC000[casa] SPS00[de] DA00[el] NC000[playa]
Simplificado género	DA0S0[el] NCS000[casa] SPS00[de] DA0S0[el] NCS000[playa]
Simplificado número	DA0F0[el] NCF000[casa] SPS00[de] DA0FS[el] NCF000[playa]

7.2. Selección de características

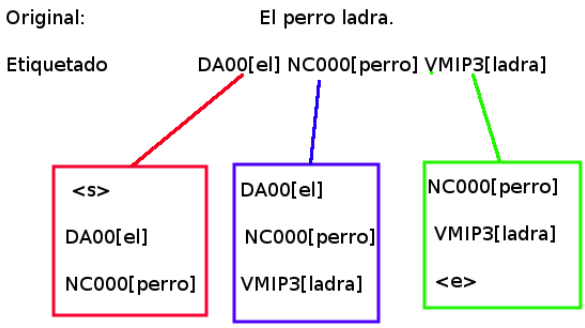
Con los modelos generados anteriores y habiendo comprobado que es posible mejorar los resultados del sistema de referencia hemos de decidir como representaremos el texto para que los distintos sistemas de clasificación reciban información relevante para la tarea. Diversas estrategias han sido utilizadas entre las que podemos diferencias aquellas en las que utilizábamos información bilingüe utilizando tanto información del lenguaje original como del lenguaje objetivo y aquellas basadas en sólo uno de ellos.

Como veremos en el apartado de resultados, el método seleccionado tras entrenar diversos clasificadores ha sido el modelo basado en ventanas de tamaño fije utilizando información monolingüe del texto castellano.

Definimos ventana como una lista de palabras de tamaño n en el cual el elemento central es aquel que queremos clasificar. El resto de la ventana contiene las palabras que preceden y siguen a la palabra en la frase manteniendo su orden.

Añadimos los caracteres especiales $<s>$ y $<e>$ para marcar el inicio y el final de la frase respectivamente, para asegurar que todas las ventanas tengan el mismo tamaño. En la figura 7.1 podemos ver un ejemplo de ventanas aplicadas a una oración:

Figura 7.2: Ejemplo de representación en ventanas



El objetivo de este método es que los clasificadores utilicen únicamente el contexto de la palabra en la oración para predecir los resultados.

Esta representación nos presenta diversas ventajas sobre otras alternativas:

- Al utilizar únicamente características del castellano evitamos errores producidos por el alineamiento automático.

- Nos proporciona un parámetro adicional que podemos ajustar en nuestros modelos. Distintos atributos a clasificar pueden beneficiarse de tener información de un contexto amplio de la palabra mientras que para otros puede generar ruido que empeore los resultados.

- El chino es un idioma en el que la información morfológica no está representada por lo que es complicado ajustar los modelos ya que para dos entradas idénticas podemos tener dos resultados distintos.

Una vez separado el texto en ventanas hemos de representarlas de forma numérica para utilizarlas posteriormente. Para ello ordenamos las palabras de nuestro corpus por número de apariciones y seleccionamos las m primeras palabras como nuestro vocabulario. Donde m es un parámetro a elegir por el usuario y que también tiene un impacto notable sobre la clasificación. Finalmente a cada palabra la representamos con su índice en el vocabulario.

Este método nos permite entrenar el modelo con valores desconocidos, tal y como los encontraría al clasificar un texto diferente, manteniendo aquellas más comunes y con más impacto en los resultados.

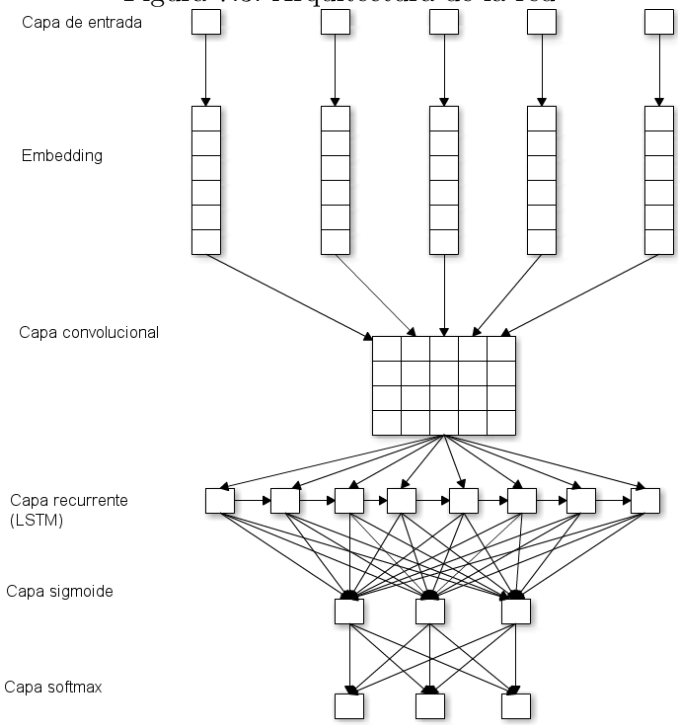
7.3. Clasificación

Diversos experimentos han sido llevados a cabo durante la realización de nuestro proyecto con la intención de encontrar el algoritmo de clasificación que nos proporcionara los mejores resultados.

La arquitectura finalmente desarrollada es una combinación de las arquitecturas propuestas en diversos artículos. La estructura de la red está fundamentada en la propuesta en el artículo *Natural language processing (almost) from scratch*[Collobert et al., 2011] en el cual se muestran dos arquitecturas distintas. En la primera se utilizan ventanas de texto y en la segunda se utilizan sintagmas y se añade el uso de una red convolucional. La red que presentamos a continuación combina ambas aproximaciones.

A su vez, al utilizar palabras simplificadas la red ha de obtener gran parte de la información a partir del orden de la secuencia de palabras que recibe. Teniendo en cuenta este aspecto y basándonos en los resultados obtenidos en generación de texto utilizando redes recurrentes [Sutskever et al., 2011, Graves, 2013] decidimos utilizarlas en nuestra arquitectura.

Figura 7.3: Arquitectura de la red



En primer lugar tenemos la capa de embedding, la cual nos permite a partir de la representación discreta de nuestras palabras crear una representación de las mismas en un espacio continuo de mayor dimensionalidad.

Pese a que los datos tras esta capa nos facilitan más información que las ventanas originales podemos mejorar su representación. Para ello utilizamos capa convolucional. Su cometido es a partir de la salida de la capa anterior reducir el tamaño de los datos mostrando las similitudes entre las palabras de la ventana.

Tras estos primeros pasos ya disponemos de los datos preparados para realizar la clasificación. Nuestra clasificación se basa en el contexto de la palabra dentro de la oración, por lo que decidimos utilizar una red recurrente que nos permita tratar las ventanas como secuencias en el tiempo.

Es decir, en qué orden aparecen las palabras en nuestra ventana marca el efecto que tienen sobre las otras. Podemos ver un ejemplo de ello en la frase *La gata del vecino*, suponiendo que todas las palabras pertenezcan a la misma ventana de texto, tratarlo como una secuencia nos permite representar que el impacto de *gata* sobre *la* sea mayor que el de *vecino*, minimizando el impacto del ruido en los datos.

Pero tratar los datos como secuencia no nos asegura un buen resultado por si mismo, ya que encontramos situaciones en las no deseamos que palabras encontradas anteriormente afecten a las siguientes. Un ejemplo de ello son los signos de puntación, los cuales marcan un cambio en aquello de lo que estamos hablando como puede ser en el caso de un punto, o elementos sin relación morfológica entre ellos como podemos encontrar entre comas en una enumeración.

Por ello utilizamos en esta tarea una capa *long short-term memory (LSTM)* que nos proporciona además de poder utilizar información de la secuencia de palabras, la capacidad de entrenar cuando se deben olvidar la información de las anteriores.

Tras ella tenemos una capa con activación Sigmoide con tantas unidades como clases pretendemos clasificar y acabar de ajustar los resultados proporcionados por la capa recurrente.

Y finalmente, al ser el objetivo de nuestra red proporcionar la probabilidad de nuestras palabras de pertenecer a cada una de las clases, una capa con Softmax como activación que nos normaliza la salida de forma que la suma de todas las salidas sea 1. Podemos definir esta función con la siguiente expresión, para cada clase:

$$P(x)_j = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$$ Siendo K el número total de clases a predecir.

Optimización

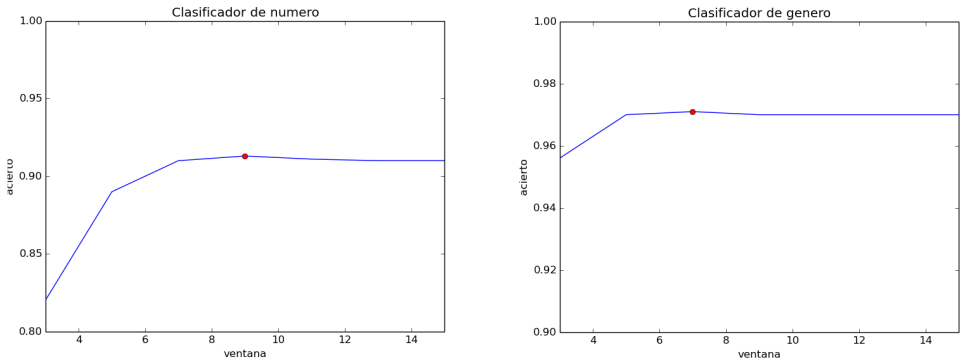
Los algoritmos de aprendizaje automático generalmente presentan distintos parámetros que es necesario ajustar a nuestros datos para conseguir un resultado óptimo. En esta sección comentaremos cuales son en nuestro sistema y que valores hemos escogido para ellos.

Tabla 7.2: Valores escogidos para los parámetros de la red.

	Número	Género
Tamaño de ventana	9	7
Tamaño de vocabulario	9000	7000
Tamaño de los *filtros*	5	5
Tamaño del *embedding*	128	128
Unidades de la capa recurrente	70	70

En primer lugar debemos empezar por el preproceso de los datos. Nuestro sistema de ventanas presenta dos parámetros que podemos modificar, el tamaño de la ventana y el tamaño del vocabulario utilizado. En la figura vemos como se comporta la red con diferentes tamaños de ventana:

Figura 7.4: Resultados de clasificación para distintos tamaños de ventana

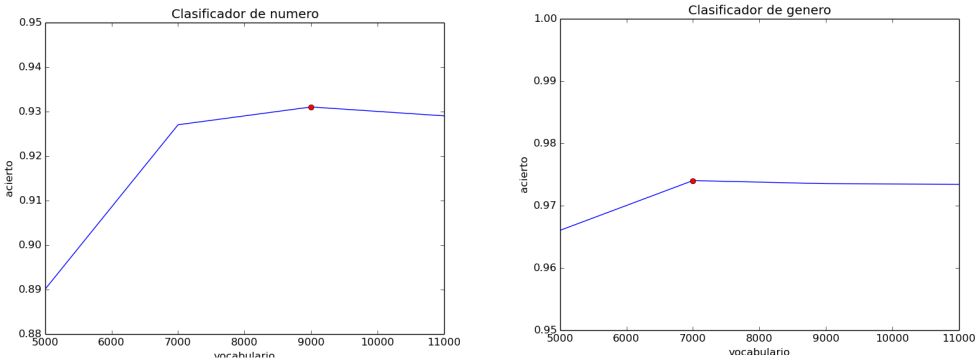


Como podemos ver en la figura si el tamaño de la ventana no es suficiente los resultados se ven afectados negativamente, mientras que cuando alcanzamos el valor óptimo, marcado en rojo, los resultados permanecen estables. Esto nos indica que un tamaño de ventana mayor no nos proporciona información relevante nueva para la tarea.

Veamos ahora el caso del vocabulario. La importancia de esta variable es que a mayor vocabulario tenemos información sobre palabras conocidas. Pero de la misma manera durante la fase de entrenamiento se ajusta menos la red para palabras desconocidas que se encuentren en textos que

no participan en el entrenamiento. En la figura vemos como varían los resultados de clasificación con distintos tamaños:

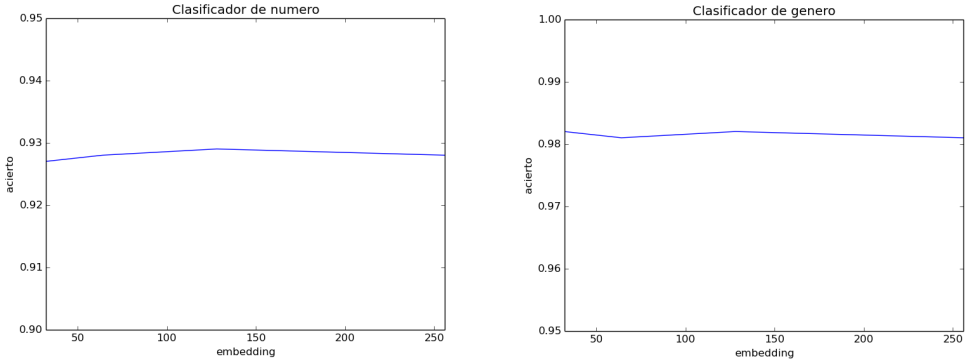
Figura 7.5: Resultados de clasificación para distintos tamaños de vocabulario



Podemos observar como tener un vocabulario mayor no nos proporciona resultados mejores cuando ya hemos alcanzado un tamaño suficiente, debido a que empezamos a no representar adecuadamente el caso de palabras desconocidas.

Una vez revisados los parámetros relacionados con el preproceso de los datos, vamos a analizar los parámetros de la red. En primer lugar vamos a analizar el tamaño del *embedding*, es decir el tamaño de los vectores que representan las palabras:

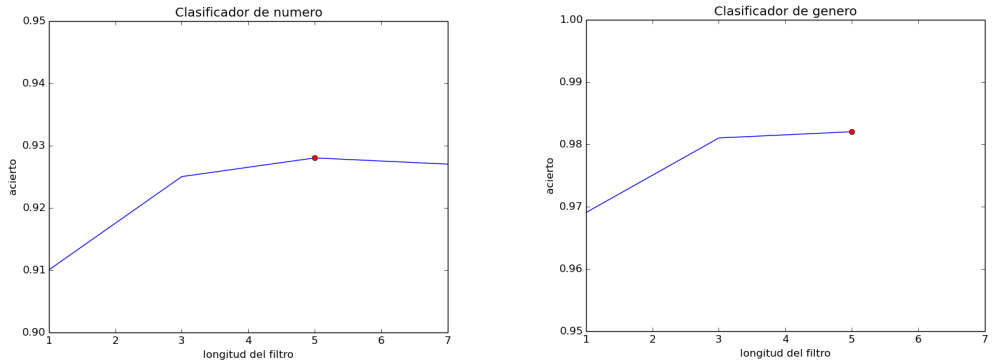
Figura 7.6: Resultados de clasificación para distintos tamaños del embedding



Como vemos no nos aporta diferencias significativas en el resultado obtenido. Decidimos utilizar un tamaño de 128 por ser el presentado en el artículo en el que se basa la red[Zheng et al., 2013].

Posteriormente, para la red convolucional hemos de ajustar el tamaño de los *filtros* que utilizaremos. Estos *filtros* al tener que multiplicarse por la salida de la capa de *embedding* están limitados a los tamaños de ventana escogidos anteriormente.

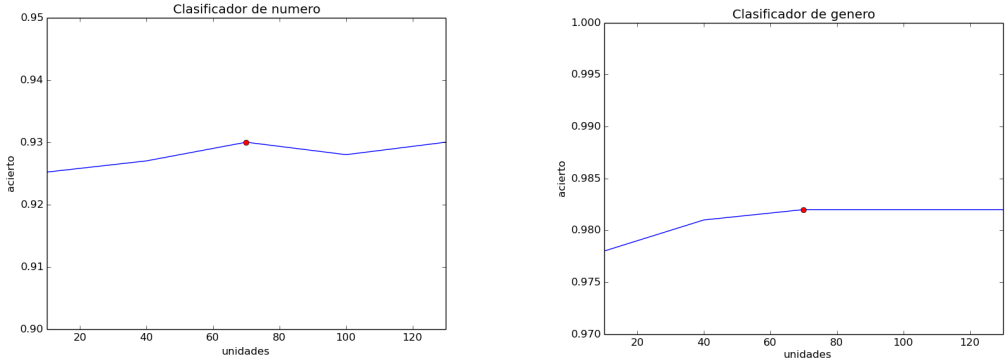
Figura 7.7: Resultados de clasificación para distintos tamaños de *filtro*



Podemos ver que ambos modelos presentan un máximo al utilizar *filtros* de 5 elementos.

Finalmente hemos de decidir el número de unidades de la red recurrente y de la capa sigmoide. Estas unidades son encargadas de aplicar los calculos a los datos y de transmitir la información a la siguiente capa o, en el caso de la red recurrente a la unidad siguiente.

Figura 7.8: Resultados de clasificación para distintos números de unidades



Como podemos ver los resultados parecen ser mejores al aumentar el número de unidades hasta que alcanzamos las 70 unidades en que se estabilizan.

7.4. Postprocesado

Con las probabilidades obtenidas por los modelos podemos generar el texto incluyendo la información morfológica.

Para ello por cada palabra del texto generamos sus etiquetas ordenadas de mayor a menor probabilidad según los resultados obtenidos. El motivo de generar todas sus etiquetas es reducir el impacto en el texto generado de las palabras que los modelos no han clasificado correctamente y que la opción más probable no existe en el idioma.

Un ejemplo de ello son palabras neutras que el modelo clasifica como masculinas o femeninas y que de solo generar esa opción sólo obtendríamos el lema como resultado.

Para realizar este proceso utilizamos los diccionarios proporcionados por *freeling*, así como sus reglas de afijos para obtener a partir de la etiqueta generada y su lema la forma final que necesitamos.
Con este proceso hemos añadido el conocimiento morfológico a nuestro modelo pero en orden de obtener mejores resultados diversas reglas han de ser añadidas:

- Las conjunciones *y* y *o* antes de vocal. Debemos controlar una vez tenemos nuestro texto generado que si encontramos una conjunción y antes de *i* o *hi* debemos substituirla por e. De la misma forma la conjunción o antes de *o* u *ho* debe ser substituida por u.

- Los verbos que tengan un pronombre como sufijo. Cuando la forma conjugada de un verbo es llana y le añadimos un sufijo, pasa a ser esdrújula y por lo tanto debemos acentuarla siempre, pese a que originalmente no.

7.5. Rescoring

Como alternativa a generar directamente el texto final final hemos experimentado con utilizar la herramienta proporcionada por *Moses* para utilizar un modelo de lenguaje junto con nuestros modelos para mejorar los resultados obtenidos.
Para ello hemos generado por cada linea del texto las n mejores alternativas según nuestros modelos por separado, en la figura 7.9 tenemos un ejemplo de ello:

Figura 7.9: Alternativas a cada frase

Original: Los niños del pueblo
Etiquetado simplificado: DA00[el] NC000[niño] SPS00+DA00[de+el] NC000[pueblo]
Género: M M N N Peso: 0,93
Número: P P N S Peso: 0.87
Etiquetado completo: DAMP00[el] NCMP000[niño] SPS00+DANN00[de+el] NCNS000[pueblo]

El problema de generar las n mejores alternativas se convierte en el problema de encontrar los n mejores caminos en un grafo pero dado que son clases estos grafos tienen ciertas características:

- Cada nodo del grafo consiste en la forma de la palabra a la que representa y su probabilidad según el modelo.
- El grafo es dirigido. Al recorrer el grafo debemos seguir el orden de la secuencia en la que se suceden las palabras en la oración.
- El peso de una arista se calcula como la probabilidad del nodo al que apunta.
- Asumiendo que las palabras que aparecen en una linea son independientes. Utilizamos como medida de la calidad de una frase la probabilidad condicionada, definida como:

$$w(X) = \prod P(X_i), \forall X_i \in X$$

Donde X es la frase a ponderar y X_i una palabra en ella.

- Para toda palabra en el grafo solo existen aristas entre ella y todas las posibilidades de la siguiente palabra en el grafo. Lo cual resulta en un grafo que presenta una capa por cada palabra y solo existen conexiones entre una capa y la siguiente, de forma similar a como se daría en una red neuronal.
- Debido a la estructura anterior el grafo siempre es acíclico.

- Al solo poder conectar una palabra con la siguiente podemos calcular el camino mínimo del grafo en tiempo en $O(n)$ ya en el caso peor tendremos que recorrer todas las capas y por cada una de ellas todas las posibilidades. Este valor podemos considerarlos constante ya que está acotado por el número de clases a predecir. En nuestro caso cada capa tendrá 3 nodos en el caso de tener información de número o uno en caso contrario.

Para generar estos caminos hemos elegido aplicar el algoritmo de Yen [Yen, 1970] substituyendo el algoritmo de Dijkstra por un recorrido de las aristas que como hemos comentado es lineal en esta implementación.

```
Data: Grafo G de la frase a elimina, K
Result: k mejores caminos de G
initialization;
A[0] = bestPath(G,0,final);
B = [] ;
i = 0;
for i  K do
    for  i in range(0, len(A[K-1])-1) do
        spurNode = A[K-1][i];
        root = A[K-1][0;i];
        for path in A do
            if root = path[0:i] then
                elimina la arista(i-1,i) de G;
            end
        end
        for node in root and node != spurNode do
            elimina el nodo node de G;
        end
        spurPath = bestPath(G,spurnode, final) totalPath = root + spurPath;
        B.append(totalPath);
        recuperar aristas de G;
        recuperar Nodos de G;
        if  B is empty then
            break;
        end
        B.sort();
        A.append(B[0]);
        B.pop();
    end
end
```

Algoritmo 1: Pseudocódigo del algoritmo de Yen

Respecto a la complejidad del algoritmo, durante la ejecución hemos de repetir el proceso de generar un nuevo camino K veces y por cada una de ellas recorrer los caminos anteriores. Podemos eliminar nodos y aristas en tiempo lineal así como calcular el mejor camino disponible. Sin embargo por cada iteración hemos de ordenar el conjunto de posibles caminos. Por lo que el coste de esta implementación, utilizando el algoritmo de *quicksort* con coste $O(NlogN)$ es $O(KN^2logN)$.

El otro aspecto a tener en cuenta es el coste en memoria del algoritmo. Las operaciones de anular y activar nodos y aristas las realizamos sobre el mismo grafo, por lo que no necesitamos espacio adicional. Sin embargo, si que necesitamos almacenar los caminos subóptimos de iteraciones anteriores. Para reducir el impacto de esta operación únicamente guardamos la secuencia de nodos recorridos de cada uno de ellos.

Con nuestras frases generadas calculamos la puntuación de cada una de ellas según modelo de lenguaje generado con *SRILM*. Las distintas opciones para cada linea juntamente con las probabilidades obtenidas por nuestros clasificadores y las obtenidas por el modelo de lenguaje se utilizan para entrenar un modelo que evalúa las probabilidades y asigna una ponderación a cada modelo para generar una nueva puntuación.

7.6. Evaluación

En este punto del proceso hemos generado una traducción completa a castellano de la entrada, pero aun no hemos comprobado que realmente suponga una mejora el proceso realizado, respecto al sistema de referencia. Para ello planteamos dos estrategias de evaluación de los resultados, automática y humana.

Evaluación automática

La evaluación automática consiste en utilizar un algoritmo que recibiendo como entrada una traducción correcta(referencia) y la generada por el sistema(hipótesis) nos de una medida de como de aproximados son los dos conjuntos de texto. En el caso de este proyecto hemos decidido utilizar la métrica *METEOR (Metric for Evaluation of Translation with Explicit Ordering)* [Denkowski and Lavie, 2014].

Este método consiste en, en primer lugar, realizar un alineamiento de la referencia y de la hipótesis, oración a oración, en el cual se tienen en cuenta palabras que figuren como sinónimas en una tabla de sinónimos, conjuntos de palabras que estén recogidos como parafraseo y palabras que compartan lema.

Para cada oración distinguimos dos tipos de palabras, las palabras de función y las palabras de contenido, por ejemplo, adverbios y preposiciones serían palabras de función mientras que los nombres pertenecerían a las palabras de contenido. Por cada par de palabras alineado m_i

calculamos cuantos palabras de función y contenido tenemos cubiertas por en la referencia ($m_i \cdot r_c$, $m_i \cdot r_f$) y en la hipótesis ($m_i \cdot h_c$, $m_i \cdot h_f$) de la siguiente forma:

$$P = \frac{\sum_i w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|}$$

$$R = \frac{\sum_i w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|}$$

Donde w_i y δ son parámetros que hemos de ajustar para garantizar resultados óptimos de traducción.

Posteriormente utilizamos los resultados anteriores para calcular la media armónica paramentrizada:

$$Fmean = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}$$

Siendo R y P los valores calculados previamente y α un parámetro a ajustar.

Para representar las palabras que aparecen en una posición incorrecta en la oración o que no aparecen calculamos un factor de penalización:

$$Pen = \gamma \cdot (\frac{ch}{m}^{\beta})$$

Siendo ch el número de oraciones, m la media de pares alineados entre la hipótesis y la referencia y β y γ parámetros a ajustar

Finalmente calculamos la puntuación final:

$$Score = (1 - Pen) \cdot Fmean$$

Evaluación humana

El proceso anterior describe como medir la calidad de la traducción automáticamente pero el objetivo final de un sistema de traducción es realizar traducciones que resulten de más utilidad a usuarios humanos. Por ello resulta interesante realizar una evaluación en la que los evaluadores sean usuarios con conocimientos del idioma que valoren los resultados obtenidos.

Para ello seleccionamos aleatoriamente n oraciones de nuestro conjunto de evaluación con sus correspondientes pares en las traducciones generadas por el sistema de referencia y por nuestra arquitectura. Al usuario se le presentan la oración correcta y las dos oraciones y ha de responder que opción considera mejor o si son iguales.

Para evitar que el evaluador pueda encontrar patrones que alteren los resultados las oraciones son presentadas como oraciones 1 y 2, y aleatoriamente para cada oración del conjunto se decide que posición ocupa cada traducción.

De esta forma recogemos los datos de los evaluadores, pero necesitamos algún método para medir si nuestros evaluadores están de acuerdo en su criterio o por el contrario los resultados se deben a cuestiones subjetivas de cada evaluador. Para medir la fiabilidad de la concordancia calculamos el valor *Fleiss Kappa* [] de los resultados obtenidos por los evaluadores.

En primer lugar calculamos la probabilidad para todos los casos evaluados de pertenecer a las clases:

$$Pj = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij}$$

En segundo lugar calculamos la concordancia entre los resultados obtenidos por los distintos evaluadores para cada oración:

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1)$$

Calculamos la media del valor obtenido para cada oración y la suma de cuadrados de las probabilidades para cada clase:

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i$$

$$\bar{P}_e = \sum_{j=1}^{k} P_j^2$$

Finalmente, calculamos el valor de *Fleiss Kappa* aplicando la siguiente expresión:

$$k = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

En la figura podemos ver algunos ejemplos de frases evaluadas. Para cada ejemplo mostramos la traducción correcta (Tr), la traducción de la referencia (Ref) y la generada utilizando nuestra arquitectura(Gen):

Tabla 7.3: Ejemplos de oraciones evaluadas

	Oraciones
1	**Tr:** b) acoge con beneplácito el fortalecimiento del mecanismo de examen entre participantes mediante la adopción de una decisión administrativa revisada sobre los exámenes entre participantes ; **Ref:** b) acoge con beneplácito por el como resultado de su administración y la decisión del mecanismo de examen entre los propios países africanos la ; **Gen:** b) acoge con beneplácito aprobadas por la debido con su decisión de examinar los administrativos , fortalecer el mecanismo de examen entre los propios países ;
2	**Tr:** observando con satisfacción que , tal como está establecido , el programa sigue sirviendo para que un mayor número de funcionarios públicos , en particular de los países en desarrollo , adquieran más conocimientos en materia de desarme , **Ref:**tomando nota con satisfacción del programa de becas de que el número cada vez mayor de conformidad con el plan de funcionarios públicos , en particular de los países en desarrollo a los funcionarios de más de desarme de conocimientos , **Gen:** tomando nota con satisfacción del programa de becas de que el número cada vez mayor de conformidad con el plan de que los funcionarios públicos en particular de los países en desarrollo tengan acceso a los funcionarios de un mayor número de los conocimientos ,
3	**Tr:** b) acoge con beneplácito el fortalecimiento del mecanismo de examen entre participantes mediante la adopción de una decisión administrativa revisada sobre los exámenes entre participantes ; **Ref:** b) acoge con beneplácito por el como resultado de su administración y la decisión del mecanismo de examen entre los propios países africanos la ; **Gen:** b) acoge con beneplácito aprobadas por la debido con su decisión de examinar los administrativos , fortalecer el mecanismo de examen entre los propios países ;
4	**Tr:** recordando que , en su 28 período de sesiones , celebrado en 1995 , la comisión decidió elaborar una legislación uniforme sobre financiación por cesión de créditos y encomendó al grupo de trabajo sobre prácticas contractuales internacionales la preparación de un proyectodocumentos oficiales de la asamblea general , quincuagésimo período de sesiones , suplemento no . 17 (a / 50 / 17) , párr . 381 . **Ref:** recordando que la comisión en su 28 período de sesiones una decisión 1995 vinculados a formular sobre la cesión de créditos con fines de los reglamentos y de que el grupo de los contratos internacionales de elaborar un proyecto de documentos oficiales de la asamblea general , en su 50 período de sesiones , suplemento no . 17 (a / 59 / 381 A/50/17) , párr . **Gen:** recordando la comisión en su 28 período de sesiones de 1995 , en la que decidió una sobre la cesión de créditos de financiación de las normas uniformes y hagan rendir cuentas a la elaboración de un contrato internacional por el grupo de los proyectos de documentos oficiales de la asamblea general , en su 50 período de sesiones , suplemento no . 17 (a / 57 / A/50/17) , párr . 381 .

En los ejemplos número 1 y 3 podemos observar como la traducción creada por el sistema de referencia presenta más errores en el orden de las palabras, a la vez que menciona a *paises africanos*, los cuales no se mencionan en la oración original. Estos problemas no se presentan en la oración generada por nuestra arquitectura.

En la segunda oración podemos ver como ambos sistemas presentan diferencias con la traducción correcta, pero al mantener mejor el orden la frase producida por nuestro sistema, el evaluador puede llegar a entender el significado de la oración original.

En el caso del ejemplo 4, podemos ver como se ha recuperado correctamente la referencia al artículo mencionado, por lo la traducción de nuestra arquitectura ha preservado parte de la información de forma más acertada que la referencia.

Capítulo 8

Fuentes de información y métodos de clasificación

En esta sección presentamos distintas alternativas a la arquitectura presentada anteriormente, que fueron planteadas durante la realización del proyecto.Principalmente nos centraremos en las fases de preprocesado y clasificación y posteriormente en la sección de resultados compararemos los resultados obtenidos.

Fuentes de información

Para poder emplear un algoritmos de aprendizaje automático sobre un conjunto de datos, éste ha de estar representado numéricamente. En la tarea que nos ocupa, en la que nos basamos en texto, no es trivial encontrar una representación que preserve la información relevante de la entrada para conseguir un buen resultado de clasificación.

Inicialmente nos planteamos utilizar la información procedente de las palabras del texto original en chino como entrada para los algoritmos. La motivación detrás de esta idea era el hecho de que el texto en chino no tiene ningún tipo de procesado, a diferencia del texto en castellano al que se le ha quitado la información morfológica.

Realizamos distintos experimentos utilizando esta información. En primer lugar se probó utilizar un vocabulario bilingüe en el que cada entrada representa una palabra del texto en castellano con su correspondiente traducción al idioma chino.

Un factor a favor de esta estrategia es que permitía en los casos en que la palabra en chino presenta diferentes flexiones debido a la morfología, pero el texto el castellano no por estar simplificado, se asignen a entradas distintas del vocabulario. Por ejemplo, para la palabra *NC000[doctor]* podemos encontrar en el texto en chino las palabras *Yïshëng* (doctor) y *Nu yïshëng* (doctora), en pinyin simplificado.

Otra estrategia empleada fue en lugar de crear ventanas de tamaño fijo, utilizar una herramienta externa que nos permita separar los sintagmas de las oraciones en chino. Para ello escogimos la herramienta *Stanford parser*, ya que es ampliamente utilizada y nos resultaba muy conveniente por incluir modelos ya entrenados.

Paralelamente experimentamos con estrategias híbridas entre la presentada en la arquitectura definitiva y las basadas únicamente en información en chino. Realizamos experimentos combinando las ventanas creadas con el texto en chino y el texto en castellano.

También se probó a añadir a las ventanas del texto en castellano información de su correspondiente par en el texto en chino. Añadimos la información sobre pronombres presentes en la oración y la longitud de la palabra chino. Podemos ver en el ejemplo anterior para la palabra doctor que la flexión de género de la palabra se realiza añadiéndole un carácter.

Todas estas implementaciones fueron finalmente descartadas debido a que dependen de la calidad del fichero de alineamiento generado por la traducción realizada con *Moses*. Analizando este fichero vemos que existen diferencias entre las traducciones correctas y los pares de palabras chino-castellano generados. Esta situación causaba que que las entradas del vocabulario no fueran correctas y que no representara la frecuencia real de aparición en el texto.

Paralelamente desarrollamos distintas variantes del sistema basado en ventanas de texto en castellano. Basándonos en el sistema propuesto en el artículo *SVMTool: A general POS tagger generator based on support vector machines* [Giménez and Màrquez, 2004] en el que se utilizaban como características ventanas de unigramas y trigramas.

Métodos de clasificación

Tal como explicamos en el apartado de planificación durante la realización del proyecto hemos realizado experimentos con distintos algoritmos de aprendizaje automático agrupándolos en tres familias: lineales, no lineales y *ensemble*.
En la primera fase de experimentación utilizamos distintos algoritmos lineales como referencia de las modificaciones que realizábamos sobre el proceso. En este punto del desarrollo aun no estaba completamente definido el preproceso que íbamos a utilizar y el algoritmo *Naive Bayes* resultaba muy conveniente como medida de la mejora que suponían los cambios introducidos por su necesitar poco tiempo de entrenamiento y no tener parámetros que ajustar.

A la vez utilizamos *support vector machines* utilizando una función de *kernel* lineal para tener una medida de los resultados del momento usando un algoritmo más sofisticado.

Con un proceso más definido y próximo al finalmente empleado comenzamos a realizar experimentos con algoritmos no lineales. Estos algoritmos son los más utilizados en tareas de etiquetado gramatical en los últimos años. Y con ellos ya pretendíamos obtener resultados que se aproximaran a los resultados el estado del arte.

Decidimos utilizar las dos familias más utilizadas para esta tarea, *support vector machines* con funciones de *kernel* no lineales y redes neuronales.

Finalmente intentamos abordar la tarea desde un enfoque distinto. Habíamos probado metódos lineales pero en los que generábamos un único modelo. Motivados por probar otra familia de algoritmos muy utilizada en tareas de clasificación, los métodos de *ensemble*. Y entre ellos el más comúnmente utilizado, *Random Forest*.

Capítulo 9

Resultados

Durante la realización de este proyecto diversos algoritmos de clasificación han sido empleados en busca del sistema que nos permitiera obtener los mejores resultados en nuestra tarea. En la siguiente tabla presentamos los resultados obtenidos para distintos algoritmos. En el caso de *Support Vector Machines (SVM)* y *Random Forest* se utilizó *10K cross validation* y para escoger los valores de los parámetros a utilizar.

Respecto a la representación de los datos todos los sistemas fueron entrenados con el mismo tratamiento de los datos, ventanas de palabras de 7 palabras y vocabulario de 7000 palabras .

Tabla 9.1: Resultados obtenidos por los diferentes algoritmos de clasificación

Algoritmo	% Género	% Número
Naive Bayes	53,5	61,3
SVM lineal	71,7	68,1
SVM cuadrático	81,3	77,8
SVM sigmoid	87,4	83,1
Random Forest	91,8	81,6
convNet+LSTM	98,4	93,7
convNet-GRU	95,1	91,4

Los datos de la figura fueron obtenidos utilizando el preproceso explicado en apartados anteriores salvo en el caso de *Random Forest* con *one hot encoding* donde en lugar de realizar el *embedding* de los datos se entrenó el sistema con la representación *one hot encoding*.

Respecto a los resultados obtenidos por *Naive Bayes* hemos de aclarar que son muy bajos debido a que al analizar su predicción, su salida para toda entrada era la clase mayoritaria en el conjunto de entrenamiento (*invariable* para género y *singular* para número).

Vemos en la tabla 9.1 como los resultados obtenidos por los clasificadores basados en redes neuronales superan a los obtenidos por el resto por lo que son los escogidos para la arquitectura

final del sistema. Observamos también que dentro de ellos, los resultados de la red *LSTM* son ligeramente mejores que los obtenidos utilizando *GRU*.

Con nuestro sistema ya completo podemos generar el texto traducido y medir los resultados obtenidos respecto a la referencia. Como para comparar nuestros resultados utilizaremos METEOR. En la figura podemos ver los resultados obtenidos para los distintos sistemas entrenados, siendo los modelos simplificados las referencias obtenidas por el sistema de traducción y los modelos generados los que produce nuestro modelo de clasificación:

Tabla 9.2: Resultados obtenidos por los modelos de referencia

Modelo	METEOR
Sistema de referencia	55,29
Simplificado Número	55,60
Simplificado Género	55,45
Simplificado	56,81

Tabla 9.3: Resultados obtenidos los modelos generados

Modelo	METEOR
Texto Generado Número	55,35
Texto Generado Género	55,39
Texto Generado	55,48
Texto con rescoring de Género y Número	54,91
Texto con rescoring de Número	**55,56**

Resulta sorprendente que el resultado utilizando *rescoring* para combinar el resultado de los modelos con un modelo de lenguaje obtenga resultados más bajos que los obtenidos con el modelo. Una causa se esta situación es que en la ponderación de los modelos realizada por la herramienta de *rescoring* de *Moses* da una ponderación de menos del 1 % al modelo de género. Esta situación en un análisis manual observamos que la traducción presenta más errores en género que la generada a partir de los modelos únicamente.

Aparentemente la herramienta asume todas las puntuaciones que introducimos son independientes, y por lo tanto no podemos forzar que asigne un único peso para las parejas de probabiliades generadas por nuestros dos modelos.

Sin embargo si únicamente realizamos el proceso de *rescoring* sobre los resultados del modelo de número, utilizando en género la probabilidad obtenida de la red, obtenemos resultados que superan a los obtenidos .
Respecto a la evaluación humana, escogimos 100 oraciones del conjunto de evaluación y pedimos a dos evaluadores humanos que decidieran cual de las oraciones propuestas les parecía una traducción más acertada. En la tabla podemos ver los resultados obtenidos:

Tabla 9.4: Resultados obtenidos en la evaluación humana

Evaluador	Sistema de referencia	Texto generado por nuestra arquitectura	Iguales
Evaluador 1	35	49	16
Evaluador 2	38	49	13

Como vemos, ambos evaluadores consideran el texto generado por nuestra arquitectura la clase que más veces proporciona la traducción más acertada.Este hecho puede ser explicado por partir de una traducción simplificada. Al ser más sencilla la tarea obtenemos mayor precisión a la hora de generar el orden de las palabras que forman la oración y al escoger la etiqueta correcta. De cara a la evaluación humana esto permite que se preserve mejor el sentido de la oración original.

Finalmente, medimos el acuerdo de las respuestas proporcionadas por los evaluadores y obtenemos una *Fleiss Kappa* $k = 0{,}6245$, lo cual nos indica que existe una alta concordancia entre las opiniones proporcionadas.

Capítulo 10

Conclusiones

En este capítulo repasaremos las conclusiones obtenidas durante la realización de este proyecto. Al margen de los resultados obtenidos, su realización a representado una gran oportunidad para descubrir el mundo de la traducción automática y la aplicación a de aprendizaje automático sobre texto, un área de estudio que supone un amplio conjunto de retos a los que nos hemos enfrentado durante estos meses.

10.1. Objetivos alcanzados

El principal objetivo de este proyecto es el desarrollo de un modelo especifico para la generación de conocimiento morfológico que presentara una mejora en ese aspecto a los resultados obtenidos por un sistema de traducción basado en frases.

A la vista de los resultados obtenidos podemos afirmar que hemos cumplido este objetivo tanto utilizando la herramienta de *rescoring* como realizando el post proceso directamente con la salida de los modelos de clasificación.

Como objetivos secundario de este proyecto nos propusimos desarrollar un clasificador que alcanzara unos resultados similares al estado del arte en tareas de etiquetado gramatical. Tal como hemos comentado en la sección de estado del arte los resultados actuales oscilan alrededor del 95 % de acierto con variaciones debidas al idioma y al conjunto de etiquetas empleado. Y como hemos visto en la sección de resultados la arquitectura propuesta en este proyecto presenta resultados muy similares, por lo que consideramos que hemos alcanzado este objetivo.

Tras realizar la evaluación humana hemos podido corroborar el planteamiento presentado en el artículo en el que fundamentamos nuestro proyecto [Costa-jussà, 2015], en el cual se defendía que realizar el traducción sobre un conjunto de información reducido mejoraba los resultados de traducción. Hemos podido observar como una vez recuperada esta información los evaluados encontraban mejoras en el orden de las palabras en las oraciones y encontraban que el significado era más cercano al del original.

10.2. Trabajo futuro

Diversos aspectos pueden continuar siendo desarrollados en este proyecto que pueden resultar de interés.

En primer lugar, tras ver que la arquitectura obtiene buenos resultados de clasificación sería interesante experimentar con su comportamiento clasificando otros aspectos de la palabra, como el tipo de palabra, el tiempo verbal, etc. Incluso si los resultados fueran positivos llegar a desarrollar un sistema de etiquetado gramatical completo.

En segundo lugar, con el objetivo de utilizar nuestro sistema con usuarios reales una posible ampliación sería implementar este sistema en el sistema web chispa.net actualmente en desarrollo.

También podría resultar de utilizad hacer un manual de uso del sistema y hacer público el sistema en alguno plataforma web como *github* para que más usuarios pudieran utilizarlo o adaptarlo a sus necesidades.

Referencias

[Atserias et al., 2006] Atserias, J., Casas, B., Comelles, E., González, M., Padró, L., and Padró, M. (2006). Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy. ELRA.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

[Boser et al., 1992] Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152.

[Carreras et al., 2004] Carreras, X., Chao, I., Padró, L., and Padró, M. (2004). Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*.

[Chen and Manning, 2014] Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

[Chollet, 2016] Chollet, F. (2016). Keras. `https://github.com/fchollet/keras`.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

[Costa-jussà, 2015] Costa-jussà, M. R. (2015). Ongoing study for enhancing chinese-spanish translation with morphology strategies. In *Proceedings of the Fourth Workshop on Hybrid Approaches to Translation (HyTra)*, pages 56–60, Beijing. Association for Computational Linguistics.

[Costa-jussà, 2015] Costa-jussà, M. R. (2015). Traducción automática entre chino y español: ¿dónde estamos? *Año VII I. Enero -Abril*, pages 16–19.

[Costa-jussà et al., 2016] Costa-jussà, M. R., España, C., Fonollosa, J., Madhyastha, P., and Escolano, C. (2016). The talp–upc spanish–english wmt biomedical task: Bilingual embeddings and char-based neural language model rescoring. In *ACL 2016 FIRST CONFERENCE ON MACHINE TRANSLATION (WMT16)*.

[Denkowski and Lavie, 2014] Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

[Española, 2009] Española, R. A. (2009). Nueva gramática de la lengua española.

[Giménez and Màrquez, 2004] Giménez, J. and Màrquez, L. (2004). Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th LREC*, Lisbon, Portugal.

[Graves, 2013] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

[Li and Cheng, 1994] Li, D. and Cheng, M. (1994). *A practical Chinese grammar for foreigners*. China Books & Periodicals.

[Padró, 2011] Padró, L. (2011). Analizadores multilingües en freeling. *Linguamatica*, 3(2):13–20.

[Padró et al., 2010] Padró, L., Collado, M., Reese, S., Lloberes, M., and Castellón, I. (2010). Freeling 2.1: Five years of open-source language processing tools. In *Proceedings of 7th Language Resources and Evaluation Conference (LREC'10)*, La Valletta, Malta.

[Padró and Stanilovsky, 2012] Padró, L. and Stanilovsky, E. (2012). Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey. ELRA.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Rafalovitch et al., 2009] Rafalovitch, A., Dale, R., et al. (2009). United nations general assembly resolutions: A six-language parallel corpus. In *Proceedings of the MT Summit*, volume 12, pages 292–299.

[Shen et al., 2014] Shen, M., Liu, H., Kawahara, D., and Kurohashi, S. (2014). Chinese morphological analysis with character-level pos tagging. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 253–258, Baltimore, Maryland. Association for Computational Linguistics.

[Socher et al., 2013] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

[Stolcke, 2002] Stolcke, A. (2002). Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286.

[Sutskever et al., 2011] Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

[Yen, 1970] Yen, J. Y. (1970). An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, pages 526–530.

[Zheng et al., 2013] Zheng, X., Chen, H., and Xu, T. (2013). Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.