Escola Tècnica Superior d'Enginyeria en Electrònica i Informàtica La Salle

Trabajo Final de Máster

Máster Universitario en Ciencia de los Datos / Data Science

TRANSCRIPCIÓN DE CHILENISMOS AL CASTELLANO MEDIANTE EL DISEÑO DE UN SISTEMA DE TRADUCCIÓN AUTOMÁTICO.

Alumno

Sebastián Larraín Velásquez

Profesora Ponente

Elisabet Golobardes i Ribé

ACTA DEL EXAMEN
DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en el día de la fecha, el alumno

Sebastián Larraín Velásquez

Expuso su Trabajo de Final de Máster, el cual trató sobre el tema siguiente:

TRANSCRIPCIÓN DE CHILENISMOS AL CASTELLANO MEDIANTE EL DISEÑO DE UN SISTEMA DE TRADUCCIÓN AUTOMÁTICO.

Acabada la exposición y contestadas por parte del alumno/a las objeciones formuladas por los miembros del tribunal, este valoró el mencionado Trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

TRANSCRIPCIÓN DE CHILENISMOS AL CASTELLANO MEDIANTE EL DISEÑO DE UN SISTEMA DE TRADUCCIÓN AUTOMÁTICO.

Autor: LARRAÍN VELÁSQUEZ, SEBASTIÁN
Asesor: GOLOBARDES I RIBÉ, ELISABET
AÑO: 2018

RESUMEN

El castellano chileno es una variante del castellano de España, que a través del tiempo ha ido adquiriendo sus propias particularidades, rasgos fonéticos, sintácticos y léxicos. La existencia de chilenismos o términos propios del español chileno, hacen que la comunicación entre chilenos y personas de habla hispanas pueda verse dificultadas al hacer uso recurrente de este tipo de palabras en una conversación coloquial.

En este trabajo se propone un sistema, que en base a una gran cantidad de texto en chileno y texto en español, obtenido desde diversas fuentes en internet (blogs, periódicos, Facebook, Twitter), pueda ser capaz de ofrecer una traducción de estos regionalismos chilenos al castellano español, a través de un sistema de traducción no supervisado, utilizando redes neuronales.

Palabras Clave: Procesamiento de Lenguaje Natural, Modelos de Lenguaje, redes neuronales recurrentes, Word embedding, Machine Learning, Inteligencia Artificial

TRANSCRIPTION OF CHILEANISMOS INTO SPANISH THROUGH THE DESIGN OF AN AUTOMATIC TRANSLATION SYSTEM.

ABSTRACT

Chilean Castilian is a variant of Spanish from Spain, which over time has acquired its own particularities, phonetic, syntactic and lexical features. The existence of Chileanisms or terms typical of Chilean Spanish, make communication between Chileans and Spanish speakers difficult because of the recurrent use of this type of words in a colloquial conversation.

This paper proposes a system, based on a large amount of text in Chilean and Spanish, obtained from various sources on the Internet (blogs, newspapers, Facebook, twitter), that may be able to offer a translation of these Chilean regionalisms into Spanish, through an unsupervised translation system, using neural networks.

Keywords: Natural Language Processing, Language Model, Word Embedding, Recurrent Neural Networks, Machine Learning.

ÍNDICE GENERAL

TABLA DE FIGURAS

1 INTRODUCCIÓN

1.1 Motivación

La principal motivación por la que nace este proyecto radica en la dificultad que existe para personas de habla hispana, comprender el dialecto chileno. El español hablado en Chile presenta sus principales diferencias frente a otros dialectos hispanos básicamente en su pronunciación, sintaxis y vocabulario. Sin embargo, la mayor dificultad que se presenta para los hispanohablantes a la hora de conversar con un chileno es comprender los chilenismos o términos propios chilenos que se usan de forma muy recurrente en esta lengua.

Por ejemplo, algunas de estas palabras son:

- **altiro o al tiro,** que quiere decir «de inmediato».
- **¿cachái?,** forma del verbo «cachar», expresión más frecuentemente usada por la juventud con el significado de «¿comprendes?», «¿entiendes?» o «¿ves?».
- **de repente,** locución propia del español, pero que en Chile adopta también el significado de «a veces» o «posiblemente».
- **fome**, voz que quiere decir «aburrido» o «sin gracia».

En internet basta con realizar una búsqueda de "*como hablan los chilenos*" o "*castellano chileno*" para obtener cientos de resultados enseñando que significan algunas frases o modismos.

Figura 1 Artículo de cómo entender el español chileno publicado en sitio www.intriper.com (Lavinia, 2018)

La idea de generar un traductor o intérprete de chileno a español responde a la necesidad de contar con una herramienta que permita comprender que significan estos chilenismos, sin depender de que alguien ya haya realizado ese trabajo de explicar que significan cada uno de ellos.

Herramientas actuales de traducción, como *Google translate*, no cuenta con opciones de traducción para regionalismos dentro de un mismo idioma, por ejemplo, español chileno a español. Sabiendo que este es un problema que se da en todos los idiomas que se hablan en más de una región geográfica, por ejemplo: inglés americano a inglés británico, portugués brasilero a portugués de Portugal, etc. una herramienta como la propuesta en este trabajo es de importancia para facilitar la comunicación entre distintas personas que comparten un idioma común.

Actualmente, existe un gran interés en todo lo referido a lo que son los sistemas de traducción automática (*Machine Translation*), gracias al uso de técnicas de aprendizaje profundo.

En las últimas décadas, ha habido un fuerte impulso en el uso de técnicas estadísticas para el desarrollo de sistemas de traducción automática. Para la aplicación de estas técnicas a un par de idiomas dado, se requiere la disponibilidad de un corpus paralelo para dicho par, es decir, un corpus de texto en el idioma de origen y otro corpus de texto conteniendo la traducción del corpus de origen, en el idioma de destino.

La popularidad de esta técnica radica en que el desarrollo de un sistema para un par de idiomas dado puede hacerse de manera muy automática, con una reducida necesidad de trabajo experto por parte de especialistas en lingüística.

Sin embargo, para realizar un sistema de traducción chileno-español, no se cuenta con un corpus paralelo, básicamente, porque ambos son el mismo idioma español, salvo estas diferencias que corresponden a los regionalismos o chilenismos, que son los que queremos traducir.

Dado la inexistencia de este corpus paralelo, se propone un sistema que no necesita de éste, logrando así, depender de trabajos realizados anteriormente. Con esto, sólo es necesario obtener grandes volúmenes de texto en ambos lenguajes para aprender de ellos y crear un sistema de forma no supervisada, es decir, sin tener que alimentar al sistema con el texto en chileno y el texto traducido al español esperado.

De este modo, el resultado esperado para este proyecto es obtener que este sistema pueda traducir chilenismos al español de forma correcta. Adicionalmente, si los resultados son interesantes, un sistema de este tipo se puede escalar a cualquier otro par de lenguaje que compartan el mismo idioma y así obtener un sistema de traducción de regionalismos ad hoc a cada contexto.

1.2 Objetivos

En el siguiente apartado, se describe cual es el objetivo general de este trabajo, y los objetivos específicos que se deben alcanzar para llevarlo a cabo.

1.2.1 Objetivo General

Diseñar y construir un sistema de traducción automática, que sea capaz de traducir chilenismos al español, entrenado en base a texto recopilado desde internet, sin el uso de corpus paralelos.

Para la realización de este proyecto se contemplan los siguientes objetivos:

1.2.2 Objetivos Específicos.

- **Obtención de grandes volúmenes de texto en lengua chilena y lengua castellana.**

  La finalidad de este punto es poder conformar ambos corpus para poder entrenar el sistema de traducción.

  Los textos que conformen estos corpus serán descargados desde internet desde sitios que serán escogidos en base a criterios a describir en el capítulo de diseño.

  Una vez que se han descargado los textos para conformar ambos corpus, éstos se deben limpiar y procesar para la correcta ingesta de los modelos a utilizar.

- **Proceso de limpieza de los textos obtenidos.**

  La ingesta de texto con que se entrenarán las redes neuronales encargadas de obtener los vectores de palabras y de generar el modelo de lenguaje, requieren que el texto se encuentre limpio, es decir, sin caracteres especiales, sin múltiples espacios en blanco, texto en minúsculas, entre otros.

  El proceso de limpieza del texto y su preprocesamiento será detallado en la sección de diseño.

- **Representación de palabras como vectores.**

  Dado que las computadoras no entienden directamente que es lo que dicen o significan las palabras, es que es necesario representarlas como vectores.

  Para realizar esto, existen diversos mecanismos, entre los que se cuentan one-hot encoding y Word embedding.

  En la sección de antecedentes, se discutirá de que trata cada mecanismo y se identifica el motivo por el cual se decide utilizar Word embedding para el desarrollo del proyecto.

  Como adelanto, Word embedding permite representar cada palabra como un vector de dimensión a definir, pero que usualmente varía entre 50 y 500, y que su principal ventaja radica en que palabras semánticamente similares tendrán representaciones similares o “cercanas” en el espacio vectorial. Por ejemplo, las representaciones vectoriales de “gato” y “perro” se encontrarán más cercanas, o serán más similares entre ellas, que las representaciones de “gato” y “casa”.

Luego de hacer el Word embedding con cada corpus, se tendrá un espacio de vectores para el lenguaje chileno y otro para el español.

- **Obtener mecanismo de traslación entre espacios de vectores para realizar traducción de palabras.**

  Una vez realizado el Word embedding y obtenido ambos espacios de vectores, será necesario obtener un mecanismo para trasladar una palabra desde el espacio vectorial chileno hacia el espacio vectorial español. Dado que ambos espacios vectoriales son representaciones del mismo idioma español y teniendo en cuenta las consideraciones de que son distintos lenguajes, se tiene como supuesto que la disposición geométrica de las palabras en ambas representaciones vectoriales debiese ser similar. Este punto se discute más adelante en la sección de antecedentes.

  Considerando lo anterior, si ambos espacios tienen distribuciones similares, entonces si un chilenismo se encuentra dispuesto en una zona donde, por ejemplo, se encuentran mayormente las representaciones de animales, al hacer la traslación o mapeo de ese chilenismo al espacio español, debiese caer también en la zona donde se encuentran mayormente las representaciones de animales. Con esto, se puede buscar cual es el vector español que se encuentre más próximo al punto donde ha caído el chilenismo en ese espacio y decir que esa representación de una palabra española corresponde a la palabra más similar a ese chilenismo.

  Del proceso anterior, se obtendrá un traductor de palabras chilenas a español, que mapea los chilenismos desde el espacio chileno al espacio español y luego busca las ***n*** palabras más similares o más cercanas.

Si bien, con esto se podría tener un primer sistema de traducción chileno-español, traduciendo solamente las palabras de una frase que contenga chilenismos, es deseable saber si esa traducción tendrá sentido en el idioma español.

Para evaluar si una frase o sentencia tiene sentido o no en un idioma, es decir, si es de uso regular, se utilizan los modelos de lenguaje.

- **Crear un modelo de lenguaje utilizando el corpus en castellano.**

  Un modelo de lenguaje, creado a partir de una lengua determinada, establece la probabilidad de que una frase o sentencia pertenezca a dicha lengua.
  Conocer esta probabilidad nos permite evaluar las posibles traducciones que se generen y preferir aquellas lingüísticamente correctas frente a aquellas que no lo sean.

  Existe una gran variedad de métodos para crear un modelo de lenguaje, pero para este proyecto, nos interesarán aquellas técnicas que permitan estimar un modelo de lenguaje de forma automática a partir de un corpus dado.

  Una vez obtenido el sistema traductor de palabras y modelo de lenguaje es posible avanzar al siguiente objetivo, que corresponde a la de traducción y evaluación de frases.

- **Evaluar las traducciones de chilenismos a español utilizando el mecanismo de traslación y el modelo de lenguaje.**

  El método propuesto para realizar la traducción de una frase conteniendo chilenismos, es el siguiente:

- Se ingresa una frase de largo ***X*** palabras, conteniendo al menos un chilenismo.
- Se traducen las ***Y*** palabras de la frase que correspondan a chilenismos, con Y<X.
- Se obtienen ***N*** traducciones para cada chilenismo, es decir, las ***N*** palabras más cercanas en el espacio español luego del mapeo desde el espacio chileno.
- Considerando los ***Y*** chilenismos y sus ***N*** traducciones, se generan Y^N frases debido a la combinación de sus traducciones.
- Se evalúan estas Y^N frases en el modelo de lenguaje y se retornan aquellas que presentan la mejor solución lingüísticamente, de acuerdo con el valor de la probabilidad obtenida en el modelo de lenguaje.

1.3 Estructura del trabajo

La estructura de este trabajo se separa en 3 partes fundamentalmente:

Parte 1. Comprende la presentación del tema. Contiene los siguientes puntos: Motivación, Presentación de los objetivos a alcanzar con el trabajo y Presentación de antecedentes.

Parte 2. Comprende el cómo se diseña, realiza y resultados del proyecto.

Parte 3. La parte final contiene las secciones de conclusiones, líneas de mejoras y trabajo futuro.

2 ANTECEDENTES

En este capítulo se darán a conocer los antecedentes y el estado actual de los distintos ámbitos sobre los que se sostiene este proyecto. Es decir, conocer en que consiste un sistema de traducción automática, los conceptos y mecanismos para realizar la vectorización de palabras, que es un modelo de lenguaje y como se puede obtener usando redes neuronales recurrentes.

2.1 Sistemas automáticos de traducción

La traducción automática trata el problema de convertir una secuencia de palabras en una lengua determinada (origen) en una secuencia de palabras distinta en otra lengua (destino), de tal forma que una sea traducción de la otra.

Un sistema de traducción automático está compuesto de 2 etapas:

- **Entrenamiento supervisado**: en esta fase el sistema recibe un corpus etiquetado, de tal forma que para cada entrada se sabe cuál debe ser la salida, y se estiman los parámetros de la red necesarios. Estos servirán durante el proceso de búsqueda para obtener la respuesta ante entradas cuya salida sea desconocida.

- **Evaluación**: en esta fase el sistema recibe una muestra y produce la salida estadísticamente más probable. Es posible medir el error cometido por el sistema si disponemos de muestras etiquetadas en esta fase.

En la actualidad, las grandes compañías, como Google, Facebook, IBM están trabajando en nuevos métodos de lenguaje automático, intentando independizar su desarrollo del uso de corpus etiquetados.

En este trabajo, no se utilizará un corpus etiquetado, sino que se recreará un corpus etiqueta para la etapa de traslación desde un espacio vectorial a otro, con el fin de traducir chilenismos.

2.2 Representación de palabras como vectores

La representación de vectores como palabra es necesario para realizar cualquier tarea que implique el manejo de texto, por ejemplo: clasificación, traducción, análisis de sentimiento, etc.

Sin embargo, no todos los mecanismos de vectorización ofrecen los mismos resultados, por lo que a continuación se discuten los métodos One-hot encoding y Word Embedding.

2.2.1 One-hot Encoding

Un vector vectorizado con el método one-hot encoding es un vector de tamaño |V|, con V igual al número de palabras que contiene el diccionario.

Cada componente del vector identifica una palabra del vocabulario, es decir, el vector one-hot de una palabra tiene un 1 en la posición que identifica dicha palabra y los |V|-1 elementos restantes son 0. Un ejemplo ilustrativo para un vocabulario formado por cuatro palabras V = {Móvil, Perro, Gato, Rojo} se muestra en la siguiente tabla:

Palabra	Vector			
Móvil	1	0	0	0
Perro	0	1	0	0
Gato	0	0	1	0
Rojo	0	0	0	1

Tabla 1 Ejemplo de representación one-hot encoding

El problema que presenta el tipo de representación *one-hot* es que no aporta información acerca de relaciones entre palabras, por ejemplo, la distancia entre los vectores “perro” y “gato” es la misma que existe entre “perro” y “móvil”, sin embargo, dado que “perro” y “gato” son ambos animales domésticos, se desearía que se encuentren a menor distancia entre ellos, que sean más “similares”. Es decir, este tipo de codificación no tiene en cuenta información acerca del contexto de las palabras que representan.

Además, one-hot encoding presenta problemas de dimensionalidad cuando se trabaja con un tamaño de vocabulario grande, ya que la dimensión de los vectores es |V|.

Esta representación contiene un gran número de ceros que obliga a trabajar con matrices que pueden generar un gran coste computacional, limitando su uso a los recursos computacionales con que se disponga.

Para solucionar estos problemas, se presenta a continuación el concepto de *Word embedding.*

2.2.2 Word Embedding

De acuerdo con Wikipedia, “*Word embedding es el nombre de un conjunto de lenguajes de modelado y técnicas de aprendizaje en procesamiento del lenguaje natural (PLN) en dónde las palabras o frases del vocabulario son vinculadas a vectores de números reales. Conceptualmente implica el encaje matemático de un espacio con una dimensión por palabra a un espacio vectorial continuo con menos dimensiones.* (Wikipedia, 2018)

El objetivo del Word Embedding es encontrar una función ***f*** que sea capaz de convertir cada palabra en un vector, de forma que la proximidad entre vectores sea equivalente a la proximidad semántica de las palabras.

Una de las cosas interesantes de esta representación vectorial es que se pueden extraer características tan relevantes como propiedades sintácticas y semánticas de las palabras. (Turian, Ratinov, & Bengio, 2010)

Uno de los modelos más usados de representaciones distribuidas de palabras es word2vec, creado en 2013 por Tomas Mikolov en Google, el que se describe a continuación.

2.2.2.1 Word2vec

Word2vec es una herramienta que sirve para el cálculo de la representación de palabras distribuidas y continuas. Fue presentado en el artículo "*Efficient Estimation of Word Representations in Vector Space*" (Mikolov, Chen, Corrado, & Dean, 2013). En este artículo, proponen dos nuevas arquitecturas de modelos para calcular representaciones vectoriales continuas de palabras a partir de conjuntos de datos de gran volumen.

Corresponde a una red neuronal de dos capas que, dada una entrada (un corpus) no etiquetada de entrenamiento, produce un vector para cada palabra, que codifica su información semántica o contextual. El propósito y utilidad de Word2vec es agrupar los vectores de palabras similares en un espacio vectorial, con lo que se puede medir la similitud semántica entre dos palabras, calculando la distancia entre sus vectores de palabra correspondientes.

Dados suficientes datos, Word2vec puede hacer suposiciones muy precisas acerca del significado de una palabra basado en apariciones pasadas. Esas suposiciones se pueden usar para establecer la asociación de una palabra con otras, por medio de operaciones lineales entre los vectores de dichas palabras.

Por ejemplo, el resultado de un cálculo de vectores ("Madrid") - ("España") + ("Francia") debiese producir como resultado ("Paris").

Las dos arquitecturas en que se basa Word2vec corresponden al modelo continuo de bolsa de palabras (CBOW) y el modelo Skip-gram

2.2.2.1.1 CBOW

El modelo CBOW busca obtener la representación (o predicción) de una palabra central a partir de las palabras que se sitúan alrededor de ella (contexto), por ejemplo, dada la frase «El perro juega con la pelota» y dada la palabra «juega» lo que se busca con el modelo CBOW es que el contexto pueda obtener una representación (o predicción) de la palabra «juega» a partir de las palabras {El, perro, con, la, pelota}.

El objetivo de CBOW es maximizar P(*palabra_objetivo* | *contexto*) a través del conjunto de entrenamiento. Esta probabilidad resulta proporcional a la distancia entre los vectores actuales asignados a la palabra objetivo y el contexto.

Este modelo se entrena un ejemplo a la vez, por lo que el objetivo en cada iteración es minimizar la distancia entre los vectores actuales para el contexto y la palabra objetivo. Al repetir este proceso en el conjunto de entrenamiento se observa que los vectores para las palabras que regularmente co-ocurren tienden a estar más cerca entre sí, y este proceso gradualmente converge a un estado final de los vectores.

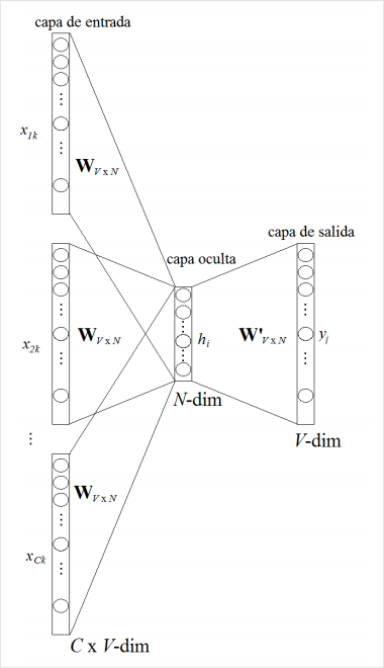


Figura 2 Modelo CBOW para Word2vec

Dado un vocabulario inicial V = $w^{(1)}$, $w^{(2)}$,….., $w^{(m)}$ con |V| = m, es decir, el vocabulario consta de m palabras. Cada palabra del vocabulario $w^{(i)}$, se representa como un vector one-hot $x^{(i)}$ R^m. Los parámetros que necesita aprender el modelo corresponden a dos matrices, $W^{(1)}$, R^{mxn} y $W^{(2)}$ R^{nxm} , donde *n* corresponde a la dimensión final que se desea para los vectores aprendidos (usualmente varía entre 50 y 500.).

La matriz $W^{(1)}$ está formada por m filas $u^{(i)}$ R^n . Estas filas representan el vector codificado de entrada de la palabra $w^{(i)}$. A su vez, la matriz de salida $W^{(2)}$ se conforma de m columnas $v^{(j)}$ R^n, que representan el vector codificado de salida de la palabra $w^{(j)}$. Es decir, por cada palabra del vocabulario, se aprenden dos vectores de palabras.

El modelo de CBOW trabaja en los siguientes pasos:

- Se desea obtener la palabra central $w^{(i)}$ dado una ventana de contexto C. ($w^{(i-C)}$, . . . , $w^{(i-1)}$, $w^{(i+1)}$, . . . , $w^{(i+C)}$)

- Se generan los vectores one-hot para cada una de las palabras del contexto ($x^{(i-C)}$, . . . , $x^{(i-1)}$, $x^{(i+1)}$,. . . , $x^{(i+C)}$).

- Se obtienen los vectores codificados de entrada por cada vector one-hot

$$u^{(i-C)} = x^{(i-C)T}W^{(1)}$$

$$u^{(i-1)} = x^{(i-1)T}W^{(1)}$$

$$u^{(i+1)} = x^{(i+1)T}W^{(1)}$$

$$u^{(i+C)} = x^{(i+C)T}W^{(1)}$$

- Se calcula la media de los vectores para obtener un vector

- Obtenemos un vector de puntaje $z = h^TW^{(2)}$.
- Se transforma en un vector de probabilidades mediante

- Se desea que las probabilidades sean iguales a las probabilidades correspondientes $y^{(i)}$, es decir, el vector one-hot de la palabra $w^{(i)}$.

De acuerdo a (Cardellino, 2015) para aprender las matrices $W^{(1)}$ y $W^{(2)}$ se define una función a minimizar basada en la entropía cruzada y se utiliza descenso por gradiente como técnica de minimización.

2.2.2.1.2 Skip-gram

Otra aproximación similar a CBOW, también presentada por Mikolov, es el modelo de Skip-gram. A diferencia de CBOW, acá se busca, dada una palabra central, por ejemplo, “sentó”, predecir las palabras que conforman su alrededor o vecindad, es decir, {“El”, “gato”, “se”, “sobre”, “el”, “sillón”}). En este caso, la palabra central es llamada contexto.

La configuración de este modelo es muy similar a la de CBOW, simplemente se cambian los roles de los vectores codificados de entrada (que pasa a ser uno solo) y los vectores codificados de salida (que pasan a ser varios). Las matrices siguen siendo las mismas.

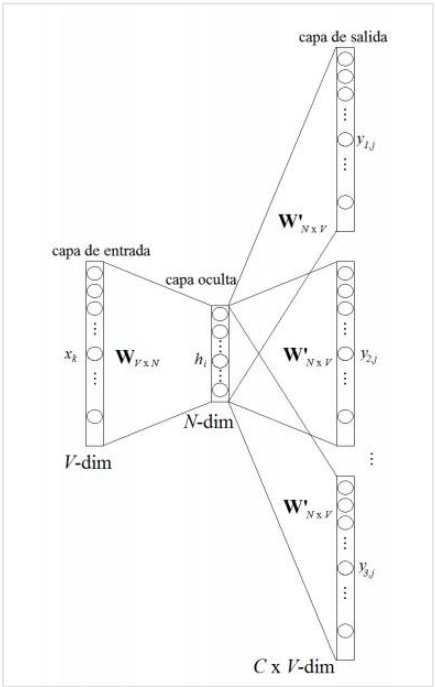


Figura 3 Modelo Skip-gram para Word2vec

La forma de trabajar del modelo de Skip-gram, es la siguiente:

- Se desea obtener las palabras ($w^{(i-C)}$, . . . , $w^{(i-1)}$, $w^{(i+1)}$, . . . , $w^{(i+C)}$) dada la palabra central $w^{(i)}$
- Se generan los vectores one-hot de la palabra $x^{(i)}$
- Se obtiene el vector codificado de entrada $u^{(i)} = x^{(i)T}W^{(1)}$
- Como es un solo vector, no es necesario obtener la media $h=u^{(i)}$
- Se generan 2C vectores de puntaje ($z^{(i-C)}$, . . . , $z^{(i-1)}$, $z^{(i+1)}$, . . . , $z^{(i+C)}$) con $z = h^{T}W^{(2)}$.
- Se transforma cada vector de puntaje en un vector de probabilidades mediante

- Se desea que los vectores de probabilidades generados sean iguales a las probabilidades correspondientes es decir, los vectores one-hot de cada palabra de salida.

De manera similar al modelo CBOW, se deben calcular los valores de las matrices $W^{(1)}$ y $W^{(2)}$, que., de acuerdo a (Cardellino, 2015) se define una función a minimizar basada en la entropía cruzada y se utiliza descenso por gradiente como técnica de minimización.

Hemos visto cuales son las características funcionales de ambos modelos. Para efectos de este trabajo, se realizarán pruebas con las 2 arquitecturas, usando distintas dimensiones y se observará quien produce un mejor resultado en base a la evaluación de las palabras similares que se obtengan para un set de prueba.

Ya una vez que se haya definido la arquitectura o modelo a utilizar, se procede a realizar la representación de las palabras para ambos corpus. Con esto ya es posible explorar un método para realizar la traducción de un chilenismo a español mediante el mapeo o traslación de la palabra chilena al espacio español mediante una función a definir.

2.3 Traducción de palabras utilizando métodos de traslación de un espacio a otro

De acuerdo con el estudio “Exploiting Similarities among Languages for Machine Translation” (Mikolov, Tomas; Quoc, Le; Sutskever, Ilya, 2013), los autores proponen que es posible aprender una proyección lineal entre dos espacios vectoriales de distintas lenguas con el fin de poder obtener traducciones de palabras. El método propuesto consiste en dos pasos. Primero, construir las representaciones vectoriales para cada lengua por separado, usando una gran cantidad de texto. Segundo, utilizando un diccionario bilingüe entre ambas lenguas, aprender una proyección lineal entre ambos espacios.

De esta manera, se puede traducir cualquier palabra que se haya visto en los corpus monolingües proyectando su representación vectorial desde el espacio del idioma de origen hasta el espacio lingüístico de destino. Una vez obtenido el vector en el espacio de la lengua de destino, se obtiene el vector de la palabra más similar a la traducción.

Las representaciones vectoriales de las palabras se aprenden utilizando los modelos distribuidos de Skip-gram o Continuous Bag-of-Words (CBOW). Hay que recordar que estos modelos aprenden representaciones de palabras usando una arquitectura de red neural simple que apunta a predecir los vecinos de una palabra.

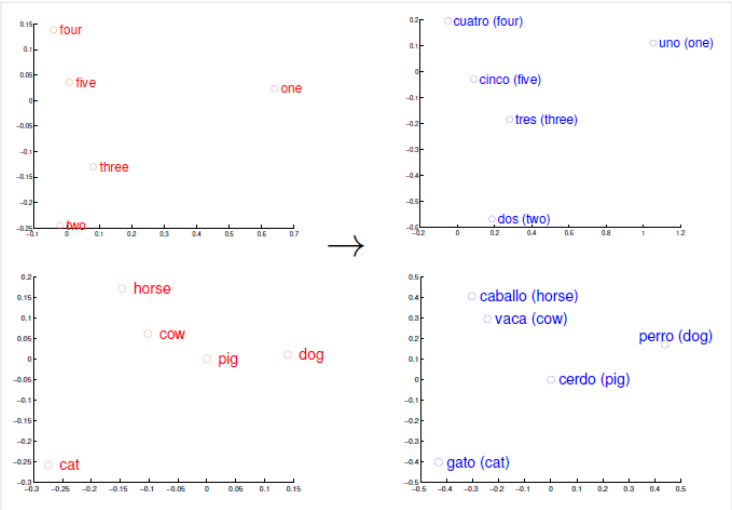


Figura 4 Representaciones vectoriales distribuidas de números y animales en inglés (izquierda) y español (derecha).

Como se aprecia en la figura 4, los vectores para números y animales en inglés y español presentan arreglos geométricos similares. La razón es que como todos los idiomas comunes comparten conceptos que están basados en el mundo real (como que el gato es, por lo general, un animal más pequeño que un perro), a menudo hay una fuerte similitud entre los espacios vectoriales.

Esta similitud de las disposiciones geométricas en los espacios vectoriales es la razón por la cual es posible aprender una proyección lineal entre espacios para producir traducciones de palabras.

Es posible visualizar que las representaciones vectoriales de palabras similares en diferentes idiomas están relacionadas por una transformación lineal. Por ejemplo, la Figura 4 muestra que los vectores para los números del uno al cinco en inglés tienen arreglos geométricos similares con los correspondientes del uno al cinco en español. La relación entre los espacios vectoriales que representan a estos dos lenguajes puede ser aprendida para producir el mapeo (es decir, una rotación y escalado). Así, si se conoce la traducción de uno y cuatro del inglés al español, podemos aprender la transformación que puede ayudarnos a traducir incluso los otros números al español.

Dado lo anterior, es necesario contar con un diccionario bilingüe previo, que sirve como puntos anclas para obtener la relación lineal entre ambos espacios.

En el caso del estudio realizado por Mikolov, para obtener los diccionarios entre idiomas, utilizan las palabras más frecuentes de los conjuntos de datos de fuentes monolingües, y han traducido estas palabras utilizando Google Translate en línea.

Considerando que el proyecto que me compete no considera traducción entre distintos idiomas, sino entre lenguajes del mismo idioma, el diccionario que se utilizará será conformado por todas aquellas palabras que se encuentren en ambos lenguajes, es decir, se realizará la intersección de los vocabularios chilenos y español, y con esas palabras se formará el diccionario necesario para obtener la relación entre espacios vectoriales. Es decir, en vez de tener un diccionario {uno:one, cuatro:four} en nuestro caso será {uno:uno, cuatro:cuatro}.

Con los antecedentes ya vistos respecto a la traducción de palabras utilizando el mapeo entre dos espacios vectoriales que contienen las representaciones de ambos lenguajes, es necesario pasar a la siguiente etapa que es conocer cómo se puede evaluar si una traducción tiene sentido o no. Para esto es necesario conocer el concepto de Modelo de Lenguaje.

2.4 Modelos de Lenguaje

Con todo lo visto hasta ahora, ya es posible obtener un sistema que traduzca chilenismos y obtener, por ejemplo, cuáles son las 3 palabras más similares en español.

Ahora, si quisiéramos traducir una frase, por ej.: *Voy altiro* , el chilenismo a traducir corresponde a la palabra "***altiro***".

Utilizando el mapeo entre ambos espacios, al pasar la representación vectorial de la palabra desde el espacio chileno al español, y obtener las 3 palabras más similares en ese espacio, debiésemos obtener resultados parecidos a: **"luego", "en seguida", "rápido"**.

Entonces, lo ideal sería evaluar cuál de las tres opciones o es la mejor traducción de la frase "Voy altiro" en el lenguaje español. Por ejemplo, dado un modelo de lenguaje entrenado con el corpus español, tenemos las siguientes probabilidades para cada frase:

P("Voy luego")=$2.4*10^{-10}$

P("Voy en seguida")= $3.2*10^{-9}$

P("Voy rápido")=$5.2*10^{-12}$

En este caso, la mayor probabilidad de que esa frase aparezca en el corpus, o se de en el lenguaje español, corresponde a **,** por lo tanto, se consideraría esa la traducción de **.**

Existen diversas maneras de obtener un modelo de lenguaje basado en un corpus mono lingual, pero para nuestro caso, nos enfocaremos en el uso de redes neuronales recurrentes (RNN), más específicamente, en una red Long Short Term Memory.

A continuación, se presenta el concepto de red neuronal feedforward, red neuronal recurrente y LSTM.

2.4.1 Red neuronal feedforward

Las redes neuronales se enmarcan en el campo del machine Learning y tienen como objetivo aproximar una función f, definiendo una aplicación g = (x; w) donde w son los pesos que se deben aprender mediante un proceso de entrenamiento. Su forma básica se conoce como feedforward, pues la información fluye desde la entrada x, pasando por g y luego pasa directamente a la salida; no existe ninguna retroalimentación, es decir, la salida de cualquier capa no afecta a esa misma capa.

Este tipo de red se suele usar para modelar relaciones entre un conjunto de variables de predicción o de entrada y una o más variables de respuesta o de salida. En otras palabras, son apropiados para cualquier problema de mapeo funcional en el que se desee saber cómo un número de variables de entrada afectan a la variable de salida.

Las redes neuronales de alimentación multicapa, también llamadas perceptrón multicapa (MLP), son el modelo de red neuronal más ampliamente estudiado y utilizado en la práctica.

Sin embargo, este tipo de red no es adecuada para tareas secuenciales, como, por ejemplo, series de tiempo, manejo de lenguaje, secuencias, etc., debido a que no poseen retroalimentación, a diferencia de las redes neuronales recurrentes, que se describen a continuación.

2.4.2 Redes neuronales recurrentes

Una red neuronal recurrente es una red neuronal que intenta modelar el comportamiento dependiente del tiempo o de la secuencia, como el idioma, los precios de las acciones, la demanda de electricidad, etc. Esto se realiza retroalimentando la salida de una capa de la red neuronal en el momento t a la entrada de la misma capa de red en el momento t + 1.

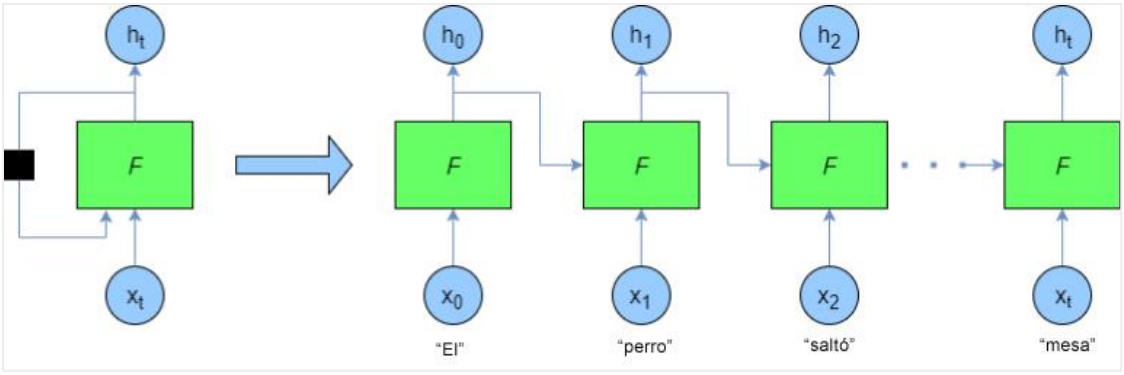


Figura 5 Ejemplo de red neuronal recurrente "desenrollada"

En la figura se observa que, en cada paso de tiempo, una nueva palabra está siendo suministrada. La salida de la **F** anterior (es decir, h_{t-1}) es suministrada a la red en cada paso de tiempo también.

En este caso de ejemplo, la frase podría ser “El perro saltó por encima de la mesa”

El problema con las redes neuronales recurrentes, construidas a partir de nodos de redes neuronales regulares, es que a medida que se intenta modelar las dependencias entre palabras o valores de secuencia que están separados por un número significativo de otras palabras, se experimenta el problema del *vanishing gradient*. Esto se debe a que los pequeños gradientes o pesos (valores inferiores a 1) se multiplican muchas veces a través de los múltiples pasos de tiempo, por lo que los gradientes se van reduciendo asintóticamente a cero. Esto significa que los pesos de esas capas anteriores no cambiarán significativamente y por lo tanto la red no aprenderá dependencias a largo plazo.

Para solucionar esto, aparecen las redes LSTM.

2.4.3 Redes Long Short Term Memory (LSTM)

Una red LSTM es una red neuronal recurrente que tiene bloques de células LSTM en lugar de capas de red neuronal estándar. El modelo fue presentado en 1997 en la publicación “LONG SHORT-TERM MEMORY” (Hochreiter & Schmidhuber, 1997).

Estas células tienen varios componentes llamados: puerta de entrada, puerta de olvido y puerta de salida. En la figura siguiente se observa una representación gráfica de la celda LSTM:

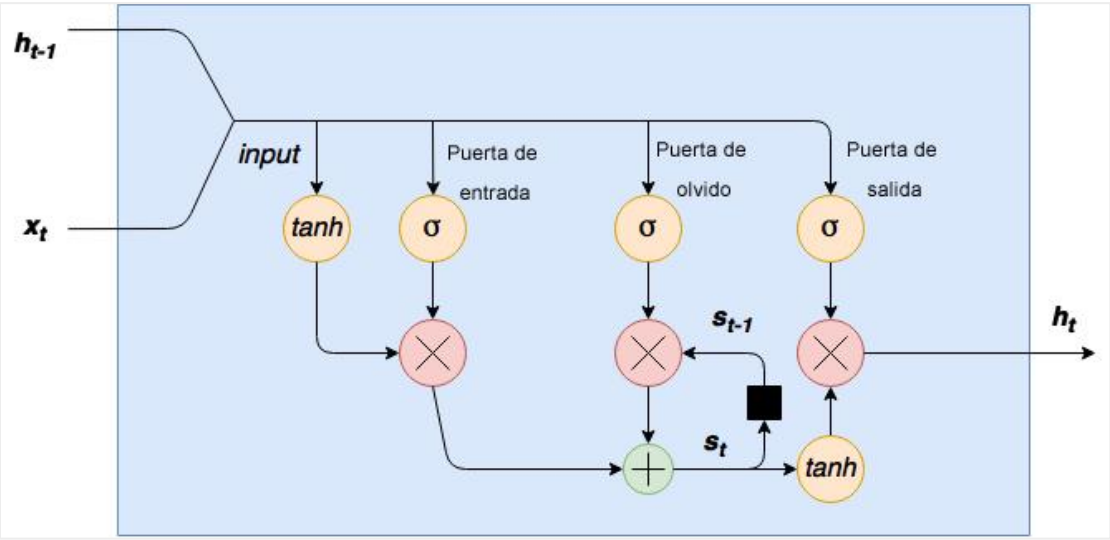


Figura 6 Esquema célula LSTM

En el lado izquierdo, se tiene la palabra/secuencia x_t siendo concatenado a la salida anterior de la celda h_{t-1}.

El primer paso para esta entrada combinada es que sea “aplastada” a través de una capa de tanh.

El segundo paso es que esta entrada pase a través de una puerta de entrada. Una puerta de entrada es una capa de nodos activados por sigmoides cuya salida se multiplica por la entrada “aplastada”. Estos sigmoides de puerta de entrada pueden actuar para "matar" cualquier elemento del vector de entrada que no sea necesario. Una función sigmoide emite valores entre 0 y 1, por lo que los pesos que conectan la entrada a estos nodos pueden ser entrenados para emitir valores cercanos a cero para "apagar" ciertos valores de entrada (o, a la inversa, salidas cercanas a 1 para "pasar a través" de otros valores).

El siguiente paso en el flujo de datos a través de esta celda es el ciclo de la puerta de olvido. Las celdas LSTM tienen una variable de estado interno s_t. Esta variable, retrasada un paso de tiempo, es decir, s_{t-1}, se añade a los datos de entrada para crear una capa efectiva de recurrencia. Esta operación de adición, en lugar de una operación de multiplicación, ayuda a reducir el riesgo de que se desvanezcan los gradientes. Sin embargo, este ciclo o loop de recurrencia es controlado por una puerta de olvido - esto funciona igual que la puerta de entrada, pero en su lugar ayuda a la red a aprender qué variables de estado deben ser "recordadas" u "olvidadas".

Por último, se tiene una función de “aplastamiento” de la capa de salida de tanh, cuya salida es controlada por una puerta de salida. Esta puerta determina qué valores son realmente permitidos como salida de la celda h_t.

El uso de redes LSTM está ampliamente difundido en tareas con datos secuenciales, por lo que se ha usado como referencia para el desarrollo de este trabajo.

En este capítulo se han descrito los antecedentes de los tópicos que se van a tratar durante el desarrollo del proyecto.

En el siguiente capítulo de describen las herramientas y el software que se utilizará para el desarrollo del proyecto.

3 ENTORNO DE TRABAJO Y HERRAMIENTAS

Para el desarrollo de este trabajo, se ha escogido el lenguaje de programación python. Durante el transcurso del Máster Universitario de Data Science, se tuvo la oportunidad de trabajar tanto con R y Python. Ambos son lenguajes de programación populares para trabajos relacionados con la ciencia de datos, que cuentan con librerías que cubren todas las necesidades de las etapas que tiene este proyecto. Por lo tanto, la decisión de escoger Python ha sido por la familiaridad con el lenguaje.

Se ha utilizado Python 3.6, junto al entorno de desarrollo Spyder.

De acuerdo con cada etapa del proyecto, se deben utilizar distintas librerías y herramientas.

Para la etapa de extracción de textos desde internet, se han utilizado las librerías

- **Beautiful Soap**: es una biblioteca de Python para analizar documentos HTML. Esta biblioteca crea un árbol con todos los elementos del documento y puede ser utilizado para extraer información. Por lo tanto, esta biblioteca es útil para extraer información de sitios web. (Wikipedia, 2018).

- **Selenium**: Es un entorno de pruebas de software para aplicaciones basadas en la web. Selenium provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas. Utilizado para simular actividad en páginas web, como hacer scroll para que carguen más comentarios en Facebook.

Para la etapa de limpieza y procesamiento de texto, se ha utilizado NLTK y

- **NLTK** (Natural Language Toolkit) (Bird, y otros, s.f.) es un kit de herramientas para el procesamiento de lenguaje natural, formado por un conjunto de bibliotecas y programas que se pueden descargar y utilizar en Python. NLTK fue creado con la finalidad de apoyar la investigación y la docencia en el estudio de procesamiento de lenguaje natural. Proporciona un conjunto de

herramientas de gran interés en PLN como son parsers, chunking, part-of-speech-tagging, tokenizers, classification algorithms, corpus, etc.

Para la etapa de Representación vectorial de palabras, se utilizará **word2vec.**

- **Word2vec** (ŘEHŮŘEK & SOJKA, 2010) es una herramienta integrada en la librería Gensim para obtener representación vectorial de palabras. Proporciona una implementación eficiente de los modelos CBOW y Skip-gram. Está compuesto por una suite de algoritmos que permiten generar modelos de forma simple y cargar modelos ya entrenados.

Por último, para la parte de la obtención del modelo de lenguaje, se utilizarán redes recurrentes del tipo LSTM. Para realizar esto, se utilizará la implementación de LSTM que se encuentra en Keras.

- **Keras**: es una API de redes neuronales de alto nivel, escrita en Python y capaz de funcionar sobre TensorFlow, CNTK o Theano. Fue desarrollado con el objetivo de permitir una rápida experimentación.

Además de estas librerías y herramientas, existen las de uso cotidiano, que se utilizan durante todo el desarrollo del proyecto, como lo son:

- **Pandas**: es una librería de código abierto con licencia BSD que proporciona estructuras de datos de alto rendimiento y fáciles de usar y herramientas de análisis de datos para el lenguaje de programación Python.

- **Numpy**: es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

Con todas las herramientas anteriores ha sido posible ejecutar cada objetivo propuesto. El diseño, experimentación y resultados obtenidos se describen en los siguientes capítulos.

4 DISEÑO DEL PROYECTO

En este capítulo se describe cómo se va a abordar cada objetivo específico propuesto, es decir, cuáles serán las herramientas que se van a utilizar, modelos y criterios para tener en cuenta para la etapa de experimentación.

A continuación, se enuncia cada objetivo específico definido en el capítulo anterior y su plan de acción.

4.1 Extracción de textos para formar corpus en lengua chilena y lengua castellano

El texto necesario para conformar los corpus en chileno y español serán extraídos desde internet utilizando métodos de *web scraping*. Para lo anterior, se considera el uso de script desarrollados en python 3.6, y el uso de recursos como *Selenium webdriver* y *Beautiful Soup*.

Se ha establecido que el dominio del tipo de texto que se desea utilizar es el de uso diario y coloquial, que es donde se utilizan con mayor frecuencia los chilenismos, es por esto, que los textos se extraerán de los siguientes lugares, tanto para chileno como castellano: *Twitter*, foros de opinión, sitios de noticias y *fan Pages* de Facebook.

Extracción texto chileno

Respecto a los textos en chileno, se ha establecido lo siguiente:

- **Twitter**

  Se descargarán los tweets que se encuentren georreferenciados en las siguientes localidades: Chile, Antofagasta, Concepción, Coquimbo, La Florida, Las Condes, Maipú, Ñuñoa, Providencia, Pudahuel, Puente Alto, Santiago, Temuco, Valparaíso, Viña del Mar.

  Se han escogido estos lugares por ser las zonas con mayor concentración de población en Chile, de acuerdo con los datos obtenidos de (Wikipedia, 2018)

- **Sitios de noticias**

  La elección de los sitios de noticias de la cual descargar texto es en base a la popularidad y cantidad de visitas que éstos tienen, de acuerdo con los datos encontrados en el sitio Alexa (Alexa, 2018). También, se considera el tipo de lenguaje utilizado en ellos.

  Se han escogido los siguientes sitios de noticias.

  - Radio Bío Bio Chile: (www.biobiochile.cl)
    Uso de lenguaje formal. Publican bastante contenido.

  - Periódico La Cuarta: (www.lacuarta.com):
    Se caracteriza por tener un lenguaje coloquial y sin faltas de ortografías, a diferencia de otras fuentes de texto coloquial, como Facebook, Twitter, foros.

- Periódico La Tercera: (www.latercera.com)
  Uso de lenguaje formal. Publican bastante contenido.

- Periódico Publimetro (www.publimetro.cl)
  Uso de lenguaje formal. Publican bastante contenido.

- ***Fan Pages* de sitios de noticias.**

Las fan Pages o páginas de Facebook de los sitios de noticias que se han escogido responden a que tienen gran cantidad de seguidores y cada publicación presenta gran cantidad de mensajes. Estos mensajes son los que se desean obtener, ya que se realizan en su mayoría lenguaje coloquial.

Figura 7 Fan Page de Radio Bio Bio (2.5 mil. de seguidores)

Se extraerán estos comentarios de los siguientes *fan Pages*:

- Fan Page Radio Biobío (www.facebook.com/RadioBioBio)
  Cuentan con 2.500.000 seguidores.

- Fan Page Periódico La Tercera (www.facebook.com/laterceracom)
  Cuentan con 3.500.000 seguidores.

- Fan Page Radio Cooperativa (www.facebook.com/Cooperativa)
  Cuentan con 1.700.000 seguidores.

- **Foros de opinión**

  Se ha optado por obtener los comentarios de un portal chileno que es el de mayor actividad durante los últimos años, denominado “El Antro” (www.antronio.cl). Contiene lenguaje exclusivamente chileno, con mucho uso de chilenismos, y al ser un foro de opinión, el lenguaje tiene tono conversacional.

Extracción texto español

Respecto a los textos en español, se extraerán desde los siguientes sitios:

- **Twitter**

  Se descargarán los tweets que se encuentren georreferenciados en las siguientes localidades: Barcelona, Bilbao, Canarias, Madrid, Málaga, Murcia, Palmas de Mallorca, Sevilla, Valencia y Zaragoza.

Es importante notar que se establece el filtro de idioma para que los tweets extraídos solo sean los que se encuentran en español, debido a la variedad lingüística de estas zonas.

Se han escogido estos lugares al ser las zonas con mayor concentración de población en España., de acuerdo con los datos obtenidos de (Wikipedia, 2018)

- **Sitios de noticias**

Se han escogido los siguientes sitios de noticias, por la cantidad de contenido que publican diariamente.

  - El Periódico (www.elperiodico.com/es/global/) en su versión global
  - Periódico La Vanguardia (www.lavanguardia.com)

- ***Fan Pages* de sitios de noticias.**

Los fan Pages de los sitios de noticias que se han escogido responden a que tienen gran cantidad de seguidores y cada publicación presenta gran cantidad de mensajes. Estos mensajes son los que se desean obtener, ya que se realizan en lenguaje coloquial.

Se extraerán estos comentarios de los siguientes *fan Pages*:

  - Fan Page El Periódico de Catalunya www.facebook.com/elperiodico.catalunya: Tiene 9.700.000 seguidores

  - Fan Page España Diario www.facebook.com/EspanaDiario.es: Tiene 3.100.000 seguidores

- **Fan Page Periódico La Vanguardia**
  www.facebook.com/LaVanguardia: Tiene 3.900.0000 seguidores.

- **Foros de opinión**

  Si bien el foro español con mayor cantidad de visitas y actividad es forococoches, el registro en éste se realiza a través de un sistema de invitaciones. Dado lo anterior, se optó por obtener texto del foro Burbuja.info (www.burbuja.info) el cual es un foro de economía pero que tiene apartados donde tratan todo tipo de temas, y no es necesario registro para tener acceso a los temas y sus comentarios.

De todo este proceso, se obtendrá un corpus en chileno y otro en español.

Cada línea de estos corpus corresponderá a un comentario, noticia o mensaje extraído de los sitios indicados anteriormente.

El paso siguiente, es realizar la limpieza de estos textos para luego alimentar con estos el modelo de *Word embedding.*

4.2 Limpieza y procesamiento de texto

La limpieza que se realizará a los textos obtenidos, con el fin de su posterior procesamiento, considera las siguientes etapas:

- Reemplazar símbolos de emojis realizados con caracteres, por ejemplo, reemplazar el Emoji :-) por la palabra ‘EMOTICON’
- Reemplazar enlaces a sitios web por la palabra ‘URL’
- Eliminar caracteres que no sean letras, números ni alguno de los siguientes signos de puntuación: . , : ¡?¿#@

- Reemplazar múltiples caracteres por uno solo, por ejemplo, reemplazar !!!!!! por !
- Reemplazar dígitos del 0 al 9 con su correspondiente nombre.
- Reemplazar los números mayores a 10 por el string “NUMERO_MAYOR_DE_10”
- Reemplazar signos de exclamación, interrogación, comas y puntos por sus respectivos nombres.
- Reemplazar las palabras que llevan por prefijo los signos @ y # por “MENCION” y “HASHTAG”
- Remover textos duplicados
- Reemplazar múltiples espacios por uno sólo.

Lo anterior se realiza utilizando expresiones regulares en python 3 con el módulo ***re.***

4.3 Representación de palabras como vectores

Una vez se tengan los corpus limpios, se procede a entrenar el modelo word2vec con las oraciones o sentencias del corpus.

Como cada línea del corpus corresponde a una noticia, mensaje o comentario, se debe transformar cada uno de éstos a oración.

Es decir, si, por ejemplo, se tiene la siguiente frase de una noticia:

bioluminiscentes utilizaban su brillo para evitar ser cazadas. Una advertencia que, paralelamente, los murciélagos aprendieron a asociar con la tasa de aleteo característica de estos insectos. Una señal multisensorial que beneficiaría tanto a luciérnagas como a murciélagos. "

Para alimentar el modelo, es necesario separarla por oraciones o sentencias. Se consideran oraciones todas aquellas que terminan con un signo de exclamación, interrogación o punto.

En el caso del ejemplo, al pasar por este proceso, debiese quedar de la siguiente manera:

entonces cuando los investigadores observaron cómo estos insectos bioluminiscentes utilizaban su brillo para evitar ser cazadas.

Una advertencia que, paralelamente, los murciélagos aprendieron a asociar con la tasa de aleteo característica de estos insectos.

Una señal multisensorial que beneficiaría tanto a luciérnagas como a murciélagos."

Luego de separar el corpus en oraciones, se puede procesar cada oración para convertirlo a lista de palabras y así entrenar el modelo word2vec.

Para lo anterior, se utilizará la implementación de word2vec que se encuentra en Gensim, que es una popular librería en python para la agrupación de texto, desarrollado por Radim Řehůřek. Se centra en el análisis temático automatizado a gran escala de textos no estructurados. (Gensim, 2018)

Se realizarán pruebas con los dos algoritmos que tiene implementado word2vec CBOW y Skip-Gram y en base a sus resultados se escogerá el definitivo

De esta etapa se obtendrán las representaciones de cada palabra del diccionario de cada uno de los corpus y se espera que las palabras que sean similares semánticamente presenten representaciones cercanas.

La evaluación que se puede realizar para este paso es en base a los resultados obtenidos de las pruebas de similitud de palabras, para comprobar si se cumple que la representación de palabras similares semánticamente se representa en vectores cercanos entre ellos, y otro tipo de operaciones vectoriales que se pueden realizar en este tipo de representaciones a detallar en la etapa de experimentación.

4.4 Traducción de palabras usando mecanismo de traslación entre espacio de vectores.

Una vez definida la arquitectura y obtenido las representaciones de las palabras en chileno y en castellano, se realizará una proyección en 2D utilizando PCA, de algunos casos de palabras para evaluar si existe una estructura similar entre ambos espacios vectoriales.

Se pretende explotar la similitud en la disposición geométrica entre ambos lenguajes, sobre todo considerando que son el mismo idioma, para aprender un a proyección lineal entre un espacio y otro.

Dado lo anterior, el algoritmo debe aprender como mapear vectores desde la representación del lenguaje de origen (chileno) hasta la representación del idioma de destino (español). Durante la traducción, proyecta la palabra de entrada para traducir al espacio de destino y devuelve palabras con la representación más cercana en el espacio de destino.

Para obtener este mapeo, se propone realizar una regresión lineal, donde los pares de palabras incrustadas en chileno y español con que se ingesta el modelo, serán aquellas que sean iguales en ambos idiomas, ya que comparten el mismo significado y debiesen encontrarse posicionadas de manera similar en ambos espacios.

De este modo, al usar el modelo obtenido con palabras chilenas, el modelo las traslada al espacio vectorial español y devuelve las palabras más cercanas.

Con esto, se tiene un traductor de palabras chileno a español que servirá para generar las oraciones que se intentan traducir y alimentar con ellas el modelo de lenguaje.

4.5 Modelo de Lenguaje

El modelo de lenguaje se implementará solamente para el idioma español, que es el idioma de destino de la traducción que se quiere evaluar.

El fin del modelo de lenguaje es evaluar si las traducciones que se ingresan tienen sentido, es decir, cual es la probabilidad de que esa oración exista en el lenguaje español.

Para realizar este modelo se utilizará una red LSTM (Long Short Term Memory) con una capa previa de Word embedding.

A la salida se dispone una capa con una función softmax que genera una distribución de probabilidad sobre todas las palabras del diccionario establecidas para el corpus español.

4.6 Long Short-Term Memory (LSTM)

Para obtener el modelo de lenguaje, se hará uso de una red LSTM. Se utilizará la implementación que provee la librería Keras.

La implementación que provee Keras es en base a crear una red secuencial, donde la primera capa será una capa de embedding, para obtener la representación vectorial de las palabras de entradas. Con esos vectores, se alimenta la capa LSTM, y a la salida va una capa con función de activación softmax para poder normalizar la probabilidad del vector de salida ***y***, que es la palabra esperada.

En la etapa de Experimentación se explica en mayor detalle este proceso.

4.7 Traducción de chileno a español

Una vez obtenido el modelo de lenguaje, para realizar la traducción de texto se consideran los siguientes pasos:

- Ingresar oración en chileno a traducir, de largo **n** palabras, con **X** chilenismos
- Separar la oración en palabras y obtener la traducción de cada chilenismo por separado. Se obtendrán **Y** traducciones por chilenismo, con **Y** a definir.
- Se generan $\mathbf{X^Y}$ oraciones a partir de la combinatoria de las posibles traducciones.
- Se evalúa cada una de las oraciones obtenidas en el modelo de lenguaje y se retornan aquellas que tengan el mejor resultado evaluado en el modelo.

4.8 Resumen

En este capítulo se ha descrito el plan de acción a considerar para llevar a cabo cada objetivo específico. Se han descrito los criterios de selección de cada lugar o sitio web del cual se extraerá texto y la manera de hacerlo, para obtener ambos corpus, chileno y español.

A continuación, se ha descrito la manera en que el texto de cada corpus debe ser limpiado, usando en su mayoría mecanismos de reemplazo de expresiones regulares. Luego como se debe realizar el procesamiento necesario de los textos para poder ingresarlos a las redes neuronales encargadas de realizar la representación vectorial de las palabras del corpus, como la red neuronal recurrente encargada de obtener el modelo de lenguaje en español.

Además, se ha definido el mecanismo para obtener las representaciones vectoriales de las palabras existentes en cada corpus. Se realizarán pruebas utilizando los modelos CBOW y Skip-gram de Word2vec, y se compararán sus resultados en base a un set de palabras de prueba a utilizar para obtener las similitudes que entrega cada modelo.

El siguiente paso ha sido describir la manera en que se realizará la traducción de palabras desde el chileno al español usando una regresión lineal, que mapea los puntos desde el espacio vectorial chileno al espacio vectorial español. Obtenida esta regresión, se podrán realizar pruebas usando un set de palabras para evaluar si las traducciones obtenidas tienen sentido.

A continuación, se ha descrito como se obtendrá el modelo de lenguaje español para evaluar las frases traducidas y de modo que se selecciona aquella que obtiene un mayor puntaje en este modelo. Se usará una red LSTM para obtener el modelo. Se evaluarán los modelos de lenguaje ingresando frases escritas de forma de correcta e incorrectas y se espera que el modelo asigne menor puntajes a aquellas escritas de mala manera gramatical.

Por último, se ha descrito como se realizarán las traducciones de frases desde el chileno al español, que es el objetivo general de este proyecto.

A continuación, se presenta la experimentación realizado y los resultados obtenidos para cada objetivo.

5 EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se describe como se desarrollaron cada uno de los objetivos, las pruebas realizadas y los resultados obtenidos.

5.1 Extracción de textos para formar corpus en lengua chilena y lengua castellano.

La extracción de texto se ha realizado por etapas, dependiendo de la fuente del texto. El orden en que se van describiendo indica el orden en el que fueron extraídos los textos.

5.1.1 Twitter

Como se ha descrito en la etapa de diseño, se ha utilizado una modificación del script GetOldTweets.py escrito por Jefferson Henrique (Henrique, 2018).

El script modificado permite indicar los siguientes parámetros:

- **Ubicación**: Ubicación geográfica en la cual se van a buscar los tweets para extraer.
- **Idioma**: El idioma definido por los usuarios en Twitter en su perfil.
- **Fecha**: Se indica la fecha para la cual se desean extraer los tweets. Si no se indica, extrae todo lo que encuentre.
- **Nombre archivo de salida**: nombre de salida donde guardan los archivos en formato csv.

Considerando los parámetros anteriores, se estableció obtener todos los tweets posibles de las localidades descritas en la etapa de diseño. Además, se establece que sólo descargue tweets de usuarios que tengan el idioma español definido en su perfil.

A continuación, se ve una muestra de tweets descargados por lenguaje.

Tweets chilenos

```
Chiquillos les cuento hoy fui a Pomaire y sentía q estaba en un capítulo de tierra adentro con la típica
música de recomiendo Chile de fondo


"Y lo más chistoso de esto es que a las profes les gusta caleta... Entonces no entiendo"


"#LaVozDelMonstruoEligióReyAJoeJonas "Los terroristas" shkdkagajks csm xd"


"@CarlangaUC les fascina andar pegando hachazos jajajajaja... Yo quiero telescópica y puerta intacta
jajajajaj nada de hachas ni sierras"
```

Figura 8 Ejemplo de tweets chilenos.

Se observa un lenguaje coloquial, con uso de chilenismos (“chiquillos”, “caleta”, “csm”), palabras cortadas, uso de emoticones y faltas de ortografía. Esta es la tónica de los textos obtenidos desde Twitter. Sirven para ayudar a identificar chilenismos, pero al no ser rigurosos con la escritura, todos los vicios de escritura pueden afectar a reconocer la semántica al momento de realizar la representación vectorial.

Tweets españoles.

```
"La fe no entiende de verjas...pic.twitter.com/HEUtSmUVXd"


"Buenos dias! Segundo dia de descanso con @sheei_laa_tm @PatricCarreras @MariaJaunzaras @beaagonzzalez
menuda nochecita juapas!"


"Negociacion colectiva no se ocupa de conciliación porque patronal y sindicatos son básicamente masculinos
-A. de las Heras en @salvadostv"


"Ale, una mosca menos en el mundo xd."


"En Valencia el Pp se ha comportado estos años como si fuera los Buendía y esto fuera Macondo."
```

Figura 9 Ejemplo de tweets españoles.

Del mismo modo anterior, se observa un lenguaje coloquial, con uso de emoticones, algunas faltas de ortografía, ausencia de algunos acentos, etc.

Lo interesante de extraer texto de Twitter es que es donde mejor se refleja cómo se habla en el día a día. Las personas suelen escribir en Twitter muy parecido a como hablan, por lo tanto, el uso de regionalismos está muy presente.

El punto negativo es la falta de rigurosidad al escribir. Básicamente, la gente que escribe por Twitter lo hace desde el móvil, lo que a veces puede dificultar realizar una escritura más prolija.

El efecto de tener diferentes versiones de una palabra debido a faltas de ortografía, por ejemplo, “días” y “dias”, debiese generar una representación de las palabras muy cercanas entre ellas, ya que se usan en el mismo contexto. Este es un alcance adicional de la representación vectorial de palabras, que sería identificar las distintas maneras en que se escribe una misma palabra.

Resultados obtenidos

La descarga de tweets ha generado la siguiente cantidad de textos:

Tweets chilenos	Tweets Españoles
698.361	383.018

Tabla 2 Total de tweets descargados por lenguaje.

A continuación, se describe como se ha realizado la extracción de texto desde los sitios de noticias.

5.1.2 Sitios de noticias.

Para la descarga de texto desde los sitios de noticias, se requiere el uso de la librería Beautiful Soup. El script desarrollado requiere en primer lugar que se defina una lista con los temas que cada página de éstas tiene, por ejemplo, el sitio web del periódico La Vanguardia presenta en su parte superior los siguientes temas: “Al minuto”, “Internacional”, Política”, “Opinión”, etc.

Figura 10 Ejemplo secciones de sitio web de La Vanguardia.

Una vez definida la lista de secciones que tiene el periódico, el script comienza a recorrer cada una de las secciones, obtiene todos los enlaces, definidos por la etiqueta HTML **“a”**, y extrae el texto de cada uno de estos enlaces. De este modo, cada vez que se ejecuta el script, se descargan todas las noticias que se han encontrado en cada una de las secciones.

Dado lo anterior, para obtener una cantidad importante de texto de los sitios de noticias, es necesario ejecutar este script periódicamente.

```
A sus 44 años, intenta disfrutar cada momento de la vida y no se obsesiona con volver a estar otra vez en
la cumbre


Con esta nueva herramienta, el usuario podría llegar a dictar comentarios o mensajes para amigos sin usar
las manos o sin mirar a la pantalla. Además se potenciaría la navegación por voz a través de la
aplicación.


El atractivo Pierce Brosnan tampoco se quedó atrás mientras estuvo al frente del personaje y acaparó
varias pifias en sus películas. En Goldeneye (1995), su primera incursión al servicio de su Majestad, baja
con una cuerda de color negro al fondo de una presa y en cuanto se lanza al vacío, la cuerda se tiñe de
blanco. Los villanos tampoco se libran de algún que otro patinazo y en otra persecución repleta de
destrozos, el faro derecho del coche de los malos se rompe cuando entra en un estrecho callejón. Sin
embargo, cuando sale, vemos el faro intacto.
```

Figura 11 Ejemplo de Noticias españolas descargadas.

En la figura anterior, se observa un ejemplo de noticias descargadas. Se observa que cada noticia se descarga por párrafos, la que puede contener más de una oración. Este punto es importante a tener en cuenta para la siguiente sección, dado que los modelos se deben alimentar con oraciones.

Es posible notar que, como ha de esperarse de un sitio de noticias, que el tono del texto es muy formal. Sin uso de regionalismos ni faltas de ortografía. Este tipo de texto es útil de enseñar al modelo para tener así un vocabulario de como son las palabras escritas correctamente.

Sin embargo, este tipo de texto no podrá obtener la semántica o caracterizar las palabras dentro de un uso coloquial, a diferencia de los tweets. Dado esto, es que es importante complementar el corpus obteniendo texto desde distintas fuentes para tratar de tener una visión general del uso cotidiano del lenguaje.

Resultados obtenidos

Los resultados obtenidos de la extracción de texto desde los sitios de noticias son:

Noticias chilenas	Noticias Españolas
19.529	7.668

Tabla 3 Cantidad de noticias descargadas desde sitios web de periódicos.

Cabe notar lo dicho anteriormente, que cada una de la línea corresponde a una noticia, que a su vez puede contener varias oraciones.

A continuación, se describe la descarga de los comentarios publicados en las noticias de los sitios de periódicos.

5.1.3 Comentarios publicados en las páginas de Facebook de sitios de noticias.

Las páginas de Facebook de los periódicos más populares en su mayoría tienen sobre un millón de seguidores, lo que implica que cada post que publican puede tener un gran alcance y una gran cantidad de comentarios. Estos comentarios son de valor para nuestro caso ya que hacen uso de un lenguaje coloquial, al tratarse de comentarios en una red social, y en su mayoría, no suele ser texto tan deformado gramaticalmente como el que se observa en Twitter.

Para descargar los comentarios desde las páginas de Facebook, se ha creado un script que mediante el uso de la herramienta *Selenium*, simula realizar un scroll en la página para que se carguen más comentarios y tener así la posibilidad de descargar más comentarios.

El problema que se presenta en este caso es que, *Selenium* abre un navegador web cada vez que se invoca, y en ese navegador no se puede simular realizar un scroll “infinito” para cargar todos los comentarios posibles, por temas de memoria RAM.

Dado la limitación anterior, es necesario ejecutar este script periódicamente para poder obtener comentarios desde estas páginas.

Resultados obtenidos

La cantidad de comentarios descargados desde las páginas descritas en el capítulo de diseño, se muestran en la siguiente tabla:

Comentarios chilenos	Comentarios Españoles
34.099	17.221

Tabla 4 Cantidad de comentarios descargados desde páginas de Facebook.

5.1.4 Foros de opinión

Los foros de opinión son una fuente valiosa de texto. Los comentarios expresados en los foros seleccionados son una mezcla entre lo observado en Facebook y Twitter en lo que se refiere al rigor gramatical. Es un lenguaje muy coloquial, con trato personal, es decir, que son verdaderas conversaciones entre las personas que comentan en un hilo, lo que no se da en las otras fuentes de texto. De este modo, este tipo de texto entrega una representación del uso del lenguaje más acorde al uso del lenguaje oral cotidiano.

Para realizar la descarga de texto desde estos sitios, se ha utilizado Beautiful Soup y Selenium. Beautiful Soup para obtener los enlaces en cada página y Selenium para simular la navegación desde un browser, ya que no al hacerlo directamente desde Beautiful Soup, al parecer ambos foros tienen bloqueados la extracción de datos de este

tipo de herramientas. Selenium, al simular que se está navegando desde un browser, no presenta ese problema, por lo que se ha optado seguir ese camino.

En ambos lenguajes, se han seleccionado los subforos de temas generales, que por lo general mezclan información de todo tipo, evitando así obtener texto sobre un dominio especifico, como, por ejemplo, si se descargará el texto desde un subforo de informática.

Resultados obtenidos

El total de mensajes descargados desde los foros se observa a continuación:

Mensajes foros chilenos	Mensajes foros españoles
138.247	234.977

Tabla 5 Cantidad de mensajes descargados desde los foros de opinión

Una vez descargados los textos desde los foros, ya se tiene una cantidad considerable de texto para conformar ambos corpus. En la siguiente tabla se resume las cantidades de texto y el total obtenido como la suma de todo lo descargado, en forma de mensajes, comentarios, tweets y noticias.

Fuente	Texto Chileno	Texto Español
Comentarios de Facebook	34.099	17.221
Noticias	19.529	7.668
Mensajes en foros	138.247	234.977
Tweets	698.361	383.018
TOTAL	***890.236***	***642.884***

Tabla 6 Detalle del texto obtenido en ambos lenguajes

Cabe recordad que cada línea de texto luego será separada en sentencias u oraciones. La cantidad final de oraciones disponibles para cada lenguaje se presentará en la siguiente etapa.

5.2 Limpieza y procesamiento de texto

La limpieza del texto se realizará sobre el corpus formado por la concatenación de todos los archivos obtenidos en la etapa anterior.

De acuerdo con lo descrito en el diseño de esta etapa, mediante el uso de expresiones regulares se buscar eliminar aquellos caracteres que no sean letras ni números, ni signos de puntuación, exclamación e interrogación.

Se reemplazan los escritos en forma de emoticón, con la palabra “emoticón”.

Los números entre 0 y 9 se reemplazan por su nombre y los mayores de 10 por el string “Numero_mayor_de_10”.

Luego de realizar algunos otros reemplazos, se eliminan los textos duplicados y se transforma todo a minúsculas.

Una vez que se tiene todo el texto limpio, se pasa cada línea, que recordemos, corresponden a párrafos, a oración.

Para realizar lo anterior se hace uso de la librería ***NLTK***, específicamente, de la función ***tokenize.***

Del proceso anterior obtenemos la siguiente cantidad de oraciones para alimentar el modelo de Word embedding:

Lenguaje	Cantidad de sentencias
chileno	1.719.005
español	1.855.105

Tabla 7 Cantidad final de sentencias por lenguaje.

Cabe notar que el modelo de Word embedding requiere que cada oración sea un arreglo de palabras. Esto se logra ejecutando la función *string.split()* sobre cada oración.

De esta manera, un ejemplo de cómo queda una sentencia luego de la limpieza y tokenización es el siguiente:

Original

Procesado

['actúa', 'a', 'partir', 'de', 'los', 'numero_mayor_de_10', 'minutos', 'y', 'ese', 'plazo', 'supera', 'la', 'hora', 'cuando', 'coma_oracion', 'como', 'es', 'el', 'caso', 'coma_oracion', 'se', 'han', 'ingerido', 'alimentos', 'fin_oracion']

Una vez se tiene limpio, separado por oraciones y cada oración conformada como lista de palabras, se puede alimentar al modelo de Word embedding.

5.3 Word Embedding

El modelo word2vec permite obtener representaciones vectoriales de palabras de acuerdo con su contenido semántico.

Para llevar a cabo esta etapa, se utiliza el modelo Word2vec disponible en la librería Gensim, descrita en la sección de Herramientas a utilizar.

Se han realizado pruebas con las distintas arquitecturas que tiene disponible *Word2vec*, los que son CBOW y Skip-gram.

Para evaluar cual arquitectura de red funciona mejor en nuestro caso, se evaluará cada modelo buscando la similitud que cada uno ofrece a un conjunto de palabras.

El conjunto de palabras para evaluar los modelos es: “Facebook”, “vehículo”, “religión”, “universidad”, “persona”.

Se han definido estas palabras porque son comunes para ambos corpus, y son de fácil evaluación a la hora de ver con cuales palabras similares a ellas responden los modelos.

Para comparar ambos modelos, los parámetros a utilizar son:

Dimensión: 300

Ventana de contexto: 5

Frecuencia mínima de palabras: 10

Iteraciones: 250

A continuación, se observan los resultados obtenidos al realizar la prueba de obtener las palabras más similares, o más cercanas en el espacio vectorial de cada lenguaje, para ambos lenguajes, comenzando con el chileno.

Similitud de palabras obtenidas para el modelo chileno						
Modelo	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**
Palabra	**Facebook**		**vehículo**		**religión**	
1	instagram	fb	camión	camión	ideología	iglesia
2	fb	instagram	auto	auto	doctrina	ideología
3	face	face	vehiculo	automóvil	iglesia	islam
4	whatsapp	whatsapp	bus	vehículos	religion	raza
5	tuiter	perfil	avión	vehiculo	tradición	doctrina

Tabla 8 Comparación similitudes entregadas por CBOW y Skip-Gram

Similitud de palabras obtenidas para el modelo chileno						
Modelo	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**
Palabra	**comer**		**francia**		**animal**	
1	dormir	carne	españa	alemania	hombre	maltrato
2	consumir	animales	alemania	españa	niño	animales
3	fumar	coman	brasil	bélgica	humano	humano
4	estudiar	comen	inglaterra	brasil	perro	carne
5	compra	tomen	eeuu	rusia	paco	crueldad

Tabla 9 Comparación similitudes entregadas por CBOW y Skip-Gram

Es posible notar, que existen casos en que los resultados son muy parecidos, como es el caso de las palabras **“facebook”, “francia”, “vehículo”.**

Sin embargo, en casos como la palabra **“comer”**, se aprecia como CBOW intenta entregar verbos que se supone son semánticamente parecidos, al menos, **“consumir”** puede reemplazar a **“comer”**. En cambio, Skip-gram retorna similitudes que suelen ser acompañadas por la palabra, por ejemplo, el primer resultado, que es **“carne”**, suele ir acompañada de la palabra **“comer”,** pero no así reemplazar el verbo **“comer”**.

Algo similar sucede con el caso de la palabra **“animal”**. Si bien CBOW devuelve como resultados palabras que pueden ser parecidas dentro de un contexto semántico, en Skip-

gram se repite lo anterior. **“maltrato”** es una palabra que puede ir acompañada de **“animal”**, no así reemplazarla.

Repitiendo el ejercicio anterior para el contexto del corpus español, obtenemos el siguiente resultado:

Similitud de palabras obtenidas para el modelo español						
Modelo	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**
Palabra	**facebook**		**vehículo**		**religión**	
1	instagram	fb	coche	coche	iglesia	ideología
2	fb	instagram	avión	mercedes	ideología	superstición
3	internet	página	camión	motor	tradición	cultura
4	whatsapp	twiter	piso	vehículos	doctrina	religioso
5	ask	internet	mercedes	camión	sociedad	islam

Tabla 10 Comparación similitudes entregadas por CBOW y Skip-Gram

Similitud de palabras obtenidas para el modelo español						
Modelo	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**	**CBOW**	**Skip-Gram**
Palabra	**cenar**		**francia**		**animal**	
1	cenar	cenar	alemania	alemania	niño	animales
2	dormir	beber	italia	italia	individuo	toro
3	beber	patatas	eeuu	bretaña	toro	humano
4	salir	carne	turquía	reino	torero	perro
5	trabajar	comiendo	china	grecia	feto	mineral

Tabla 11 Comparación similitudes entregadas por CBOW y Skip-Gram

De forma similar al caso anterior, existen casos en las que los resultados obtenidos son muy similares. Quizás la mayor diferencia se mantiene en el caso del verbo “comer”, donde nuevamente Skip-gram devuelve como resultado palabras que suelen ser acompañadas por “comer” como “carne” o “patata”.

Esto puede suceder debido a naturaleza de Skip-gram, ya que su función es predecir contextos en base a una palabra.

Considerando que nuestro trabajo consiste en buscar palabras que puedan funcionar como reemplazo de la palabra consultada, vemos que el modelo CBOW es el que mejor se adapta a esto, de acuerdo con los resultados recién vistos.

Para evaluar el modelo obtenido con CBOW, la representación vectorial obtenida debiese permitir realizar operaciones vectoriales del tipo:

-vector(hombre) + vector(mujer) = vector(reina).

El resultado obtenido en ambos lenguajes considerando el caso anterior se observa a continuación:

```
In [93]: model_chileno.most_similar(positive=['rey', 'mujer'], negative = ["hombre"])
Out[93]:
[('banda', 0.4910160303115845),
 ('canción', 0.47302648425102234),
 ('fiesta', 0.4711221754550934),
 ('reina', 0.4691287875175476),
 ('cancion', 0.46673429012298584),
 ('estrella', 0.458251029253006),
 ('mina', 0.45459291338920593),
 ('frase', 0.450054407119751),
 ('ciudad', 0.44918403029441833),
 ('selección', 0.44270163774490356)]
```

Figura 12 Operación vectorial en chileno

```
In [92]: model_español.most_similar(positive=['rey', 'mujer'], negative = ["hombre"])
Out[92]:
[('pareja', 0.5681624412536621),
 ('familia', 0.5403541922569275),
 ('canción', 0.522144079208374),
 ('novia', 0.521117091178894),
 ('reina', 0.491806298494339),
 ('amiga', 0.4650026857852936),
 ('tía', 0.46396100521087646),
 ('boda', 0.4619254469871521),
 ('foto', 0.4585534632205963),
 ('hermana', 0.45552849769592285)]
```

Figura 13 Operación vectorial en español

Se observa que en ambos casos aparece la palabra “reina”, además de otras palabras similares en las primeras posiciones.

Otra operación que se puede realizar con los vectores es identificar que palabra no pertenece a una lista de palabras dada. Se observan los resultados a continuación:

```
In [94]: model_chileno.doesnt_match("manzana blanco amarillo azul".split())
Out[94]: 'manzana'
In [95]: model_español.doesnt_match("manzana blanco amarillo azul".split())
Out[95]: 'manzana'
```

Figura 14 Resultado de encontrar que palabra no pertenece a la lista.

En ambos casos, la palabra que no pertenece a la lista es manzana, lo que coincide con lo entregado como resultado.

Dado todo lo visto, es posible observar que en general, la representación vectorial ha logrado capturar de buena manera la semántica que rodea a las palabras. La lista de similitudes vista en un comienzo parece entregar resultados coherentes del mismo modo que las operaciones vectoriales.

Dado esto, es posible ahora avanzar a la siguiente etapa, que trata la experimentación realizada para obtener la proyección lineal desde un espacio a otro para poder traducir palabras chilenas a español.

5.4 Realizar traducción de palabras usando mecanismo de traslación entre espacio de vectores

De acuerdo con lo visto en el estudio de Mikolov, Le, & Sutskever's (Mikolov, V. Lee, & Sutskever, Exploiting Similarities among Languages for Machine Translation, 2013) donde los autores parten de la intuición de que conceptos similares, expresados en distintos lenguajes, deben tener distribuciones geométricas similares en un espacio vectorial, entonces, es posible aprender una matriz de transformación lineal entre ambos modelos.

Con la intención de poder traducir chilenismos al español, es que se busca obtener esta relación lineal entre ambos espacios vectoriales.

Para observarlo de forma gráfica, se aplica PCA a 2 dimensiones, sobre un set de palabras en ambos modelos y se observa la distribución obtenida en el espacio vectorial.

Se muestra a continuación los casos de palabras para números del 0 al 9 y de un grupo de animales.

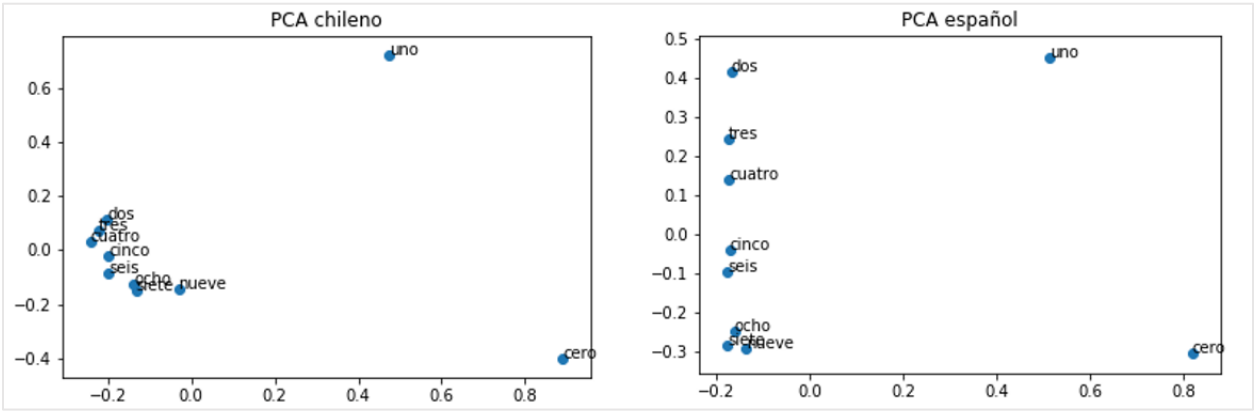


Figura 15 PCA sobre nombre de los números para comparar su posición relativa.

Se observa que, si bien la distribución en ambos espacios no es idéntica, si mantienen cierta relación.

En el caso de los números, se observa que el “uno” se mantiene en la esquina superior derecha y el “cero” en la inferior, dejando al resto de números agrupados en la zona izquierda. En este punto se observa la mayor diferencia en cuanto a la aglomeración de los números, pero cabe destacar que mantienen el mismo orden desde arriba hasta abajo.

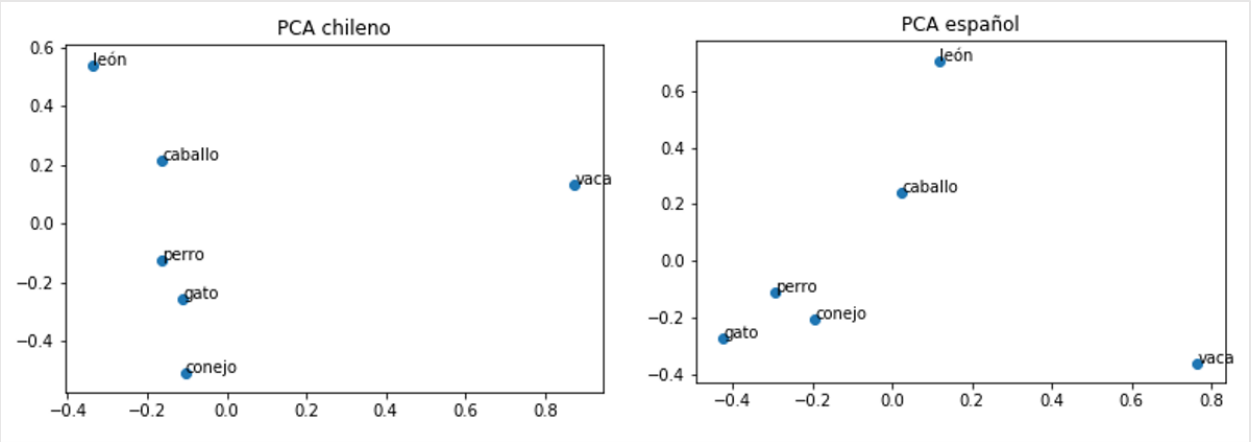


Figura 16 PCA sobre nombre animales para comparar su posición relativa.

En este caso, de forma similar que el anterior, la distribución no es exactamente la misma, pero es bastante similar, considerando una rotación entre ambos espacios.

Si se rotase el espacio español y se transforma su escala, se puede obtener una representación más similar en ambos casos, como se observa a continuación, donde la rotación y transformación se ha realizado de manera manual.

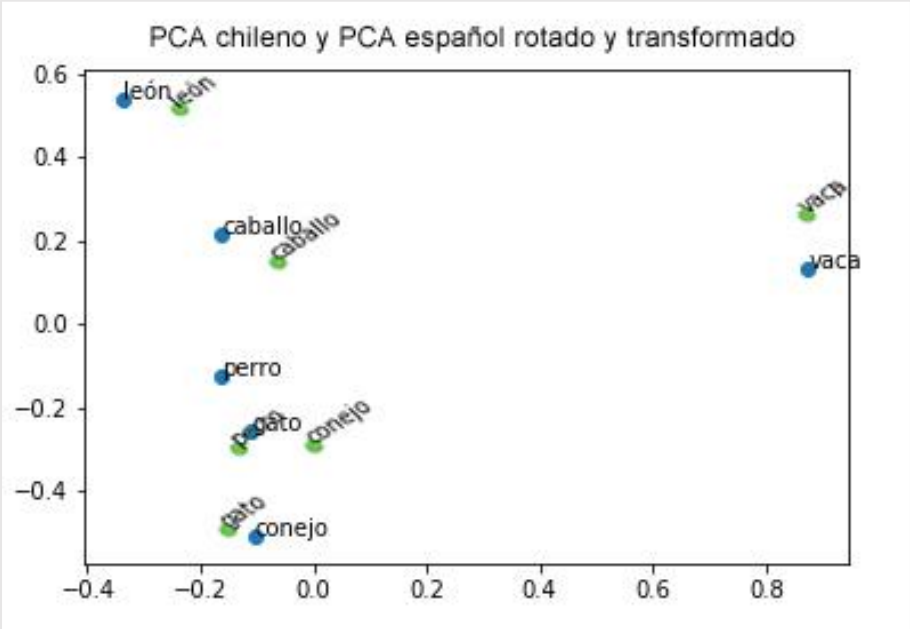


Figura 17 PCA ambos espacios. Español rotado y transformado

De acuerdo con lo anterior, y considerando el supuesto de que ambos lenguajes mantienen similitudes en su representación vectorial, se procede a encontrar la matriz de traslación que conecta ambos modelos.

Para realizar esto, se utiliza una regresión lineal donde el espacio de origen es el espacio de vectores chilenos, y el de llegada, el espacio español.

Como se ha descrito anteriormente en los antecedentes de este objetivo, el diccionario con que se entrenara el modelo de regresión se obtendrá de una lista de palabras que contengan todas aquellas que estén presente en ambos diccionarios. Esta lista se obtiene de la intersección de ambos vocabularios. De esta manera, la regresión lineal se realiza desde el espacio chileno al espacio español, debiendo realizar el ajuste sobre estas palabras que se encuentran en ambos diccionarios.

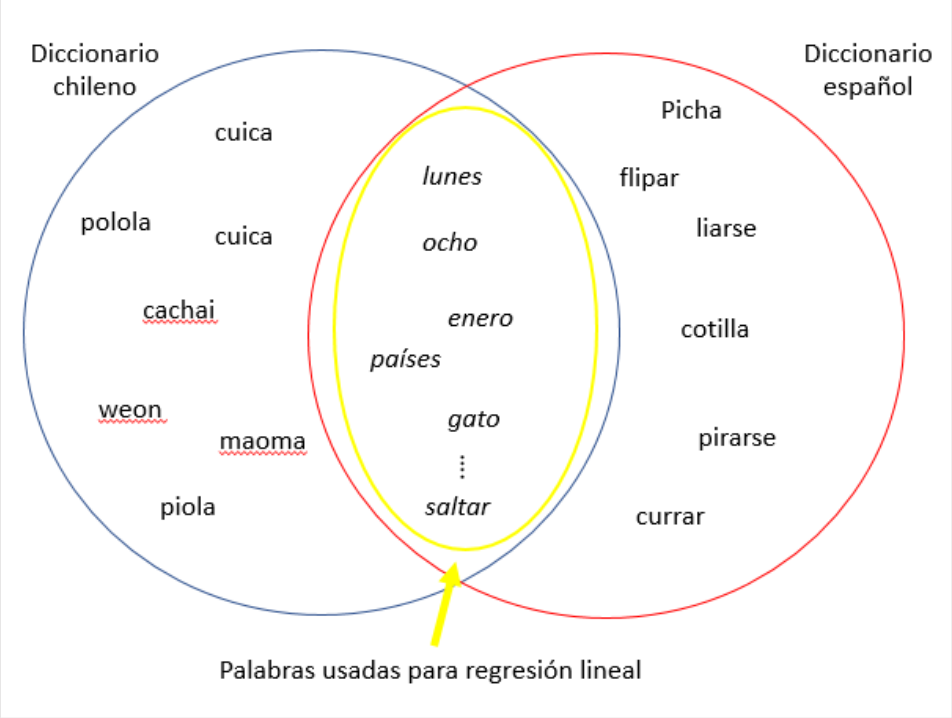


Figura 18 Intersección de palabras de ambos diccionarios para obtener aquellas a utilizar para entrenar regresión lineal.

La regresión lineal se realizará en python, utilizando el módulo *Linear Regression* de la librería *SciKit Learn.* (Scikit-learn, 2018).

Resultados

De la intersección de ambos diccionarios, se obtuvieron 37.956 palabras. Cabe destacar que el diccionario chileno cuenta con 64.541 y el español con 75.567

Es decir, se utilizarán 37.956 pares de palabras para entrenar el modelo de regresión lineal y obtener la función de traslación.

Una vez realizada la regresión lineal, se tiene un modelo que mapea una palabra desde el espacio vectorial chileno al espacio vectorial español. Es necesario luego, obtener cuales son las palabras más similares en el espacio español, utilizando la distancia coseno.

Para poder evaluar si las traducciones realizadas son correctas, se utilizarán un conjunto de chilenismos. En primer lugar, se traducirán palabras que se encuentran aceptadas en la RAE, para luego, traducir chilenismos de uso propiamente chileno, es decir, que no están incluidos en la **RAE**.

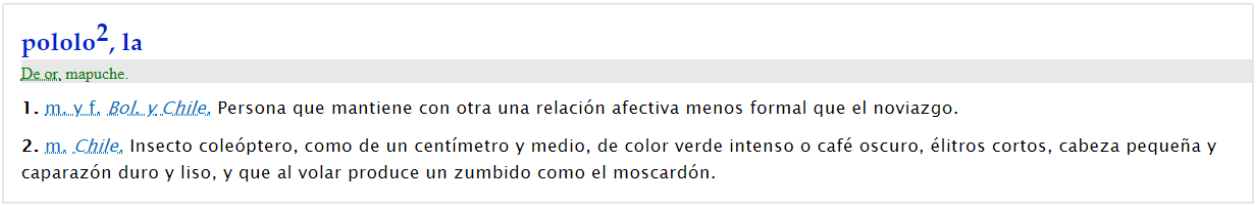
5.4.1 Traducciones de palabras incluidas en la RAE.

Traducción 1.

“POLOLO”

En el contexto chileno, pololo se refiere al novio de una persona.

Definición RAE



pololo[2], **la**

De or. mapuche.

1. m. y f. *Bol. y Chile.* Persona que mantiene con otra una relación afectiva menos formal que el noviazgo.

2. m. *Chile.* Insecto coleóptero, como de un centímetro y medio, de color verde intenso o café oscuro, élitros cortos, cabeza pequeña y caparazón duro y liso, y que al volar produce un zumbido como el moscardón.

Figura 19 Definición de "pololo" según la RAE

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	pololo
Traducción al español	novio
	perro
	niño
	hermano
	padre

Tabla 12 Traducción de "pololo"

Se observa que el primer resultado calza con la definición 1 propuesta por la RAE.

Traducción 2.

“ALTIRO”

En el contexto chileno, altiro es una palabra que se suele usar para responder que algo se va a realizar luego o rápido.

Definición RAE

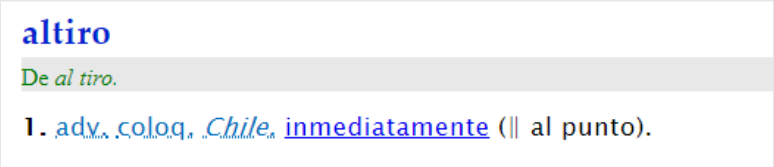


Figura 20 Definición de "altiro" según la RAE

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	altiro
Traducción al español	oye
	jajaja
	eh
	apresura
	si
	corriendo
	luego

Tabla 13 Traducción de "altiro"

Se observa que las traducciones que serían correctas no aparecen en las primeras posiciones, como, por ejemplo, *apresura* , *corriendo* y *luego* . De todos modos, aparecen entre los resultados, por lo que se considera que el algoritmo de traslación está mapeando de forma correcta las palabras desde el espacio chileno al español.

Traducción 3

“FOME”

En el contexto chileno, fome se suele utilizar para decir que algo no tiene ninguna gracia.

Definición RAE

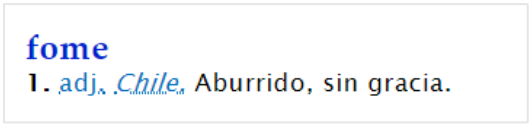


Figura 21 Definición de "fome" según la RAE

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	fome
Traducción al español	Triste
	Bonito
	guapo
	gracioso
	guapa
	bonita

Tabla 14 Traducción de "fome"

La primera traducción, “triste” se puede tomar como correcta considerando un contexto de algo falto de energía, gracia. Sin embargo, aparecen palabras que son antónimas al significado de fome, como, por ejemplo, gracioso, guapo, guapa. Quizás esto pueda deberse a que fome puede usarse como un adjetivo que no necesita mayor contexto, al igual que guapo, gracioso, etc. Por ejemplo, basta con decir “el tipo guapo”, “el tipo fome” o “el tipo gracioso”. Al entrenar la red neuronal con estas oraciones, es posible que sitúe a los 3 adjetivos en representaciones cercanas entre sí.

5.4.2 Traducciones de palabras no incluidas en la RAE.

Traducción 4

"**CACHAI**"

Es un modismo que se usa como si se tratase de la palabra "entender". Por ejemplo, "Para poder surfear, tienes que mantener el equilibrio, cachai?.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	**cachai**
Traducción al español	demuestra
	sabes
	sepas
	sabe
	es
	crees

Tabla 15 Traducción de "cachai"

En el uso diario, me parece que "sabes" corresponde a una buena aproximación de "cachai", ya que en el español se suele usar dentro de un contexto similar.

Traducción 5

"**WEÓN**"

"Weón" se podría considerar como una de las palabras nacionales de Chile. Se puede usar en distintos contextos. Puede significar tanto como "idiota" o como "amigo", pero siempre refiriéndose a alguien de sexo masculino.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	weón
Traducción al español	tío
	chico
	hombre
	niño
	tio
	chaval

Tabla 16 Traducción de "weón"

Se observa que los resultados obtenidos son acordes dentro del contexto del lenguaje español. Es común referirse a un hombre tanto como "tío", "chaval", "hombre", etc.

Traducción 6

"**AUTO**"

En chile, al automóvil se le suele decir auto. Es el equivalente a decir coche en España.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	**auto**
Traducción al español	coche
	piso
	vehículo
	edificio
	cuerpo
	sitio

Tabla 17 Traducción de "auto"

Se observa que el primer resultado es su equivalente exacto en español.

Traducción 7

"**MINA**"

En chile, se usa para como sinónimo de mujer, chica, muchacha. Además del genérico se usa para denominar a las chicas atractivas.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	mina
Traducción al español	chica
	mujer
	tía
	niña
	persona
	foto

Tabla 18 Traducción de "mina"

Se observa que los primeros resultados son acordes al uso en chileno.

Traducción 8

“BACÁN”

En chile, se usa bacán para referirse a algo que es o a estado muy bueno, excelente, genial, etc.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	bacán
Traducción al español	bonito
	guapa
	guapo
	triste
	bonita
	guay

Tabla 19 Traducción de "bacán"

Los resultados tienen sentido, salvo “triste”, que se puede deber al mismo efecto visto con la palabra “fome” en la traducción 3.

Traducción 9

“CABRO”

En chile, se usa “cabro” o “cabra” para referirse a los niños, o también a los adultos con actitudes de niños.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	cabro
Traducción al español	niño
	tío
	chico
	tio
	perro
	hombre

Tabla 20 Traducción de "cabro"

Los resultados tienen sentido en relación con el uso que se le da en Chile, salvo la palabra “perro”.

Traducción 10

“MICRO”

“Micro” es el equivalente a “autobús” en español.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	micro
Traducción al español	cama
	calle
	playa
	habitación
	piscina
	silla

Tabla 21 Traducción de "micro"

En este caso, no hay ningún resultado correcto.

Traducción 11

“POLERA”

“Polera” es el equivalente a “camiseta” en español.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	micro
Traducción al español	foto
	canción
	sonrisa
	camiseta
	camisa
	chaqueta

Tabla 22 Traducción de "polera"

Vemos que, en este caso, recién en la cuarta posición aparece camiseta.

Traducción 12

“CAHUÍN”

“Cahuín” se refiere a comentarios malintencionados que provocan discrepancias entre las personas.

Resultado obtenido de traducción chileno – español por método de traslación

Chilenismo	cahuín
Traducción al español	misterio
	tema
	experimento
	asunto
	chico
	problema

Tabla 23 Traducción de "cahuín"

Los resultados obtenidos, si bien no son traducciones literales, tampoco están tan lejos del concepto. Por ejemplo, “tema”, “asunto”, “problema”, se refieren a contexto similares a “cahuín”.

Para evaluar el modelo de traslación entre espacios para poder obtener una traducción de los chilenismos, dado que no tenemos un diccionario bilingüe, es necesario realizarlo en base al criterio de las traducciones obtenidas.

Hemos podido notar que, de las 12 traducciones evaluadas, en general, los chilenismos, al ser trasladados al espacio español suelen caer en una zona al menos semánticamente parecida. Se dan los casos en que el primer resultado es la mejor traducción posible como “auto” => “coche”, pero hay otros en que la mejor traducción aparece en posiciones más abajo, como “polera” => ”camiseta”.

Los resultados obtenidos dependen directamente de que tan bien los modelos han sido capaces de representar la semántica de las palabras en base al corpus con que se han alimentado.

La representación de las palabras dependerá exclusivamente de la cantidad de texto y el tipo de éste con que se alimenten los modelos.

En nuestro caso, podemos seguir adelante con la obtención de un modelo de lenguaje.

Este modelo será el encargado de, por ejemplo, al momento de obtener solo una de las 5 traducciones correctas, evaluar las frases formadas con esas traducciones y devolver aquella que tenga sentido en el contexto español.

5.5 Modelo de lenguaje

El modelo de lenguaje se realiza utilizando solo el corpus en español. Esto es con la intención de evaluar todas las combinaciones de oraciones que se obtendrán de las traducciones por palabra de las frases que se desean traducir y entregar aquella que tenga una mejor representación en el lenguaje español.

Para obtener el modelo de lenguaje, se utiliza la librería Keras que incluye la implementación de LSTM.

5.5.1 Implementación LSTM

Cabe recordar que LSTM es un modelo de red neuronal recurrente, al que hay que ingresarle los datos en forma secuencial. Debido a esto, se debe procesar el corpus de entrada para que quede de la forma requerida, según lo visto en el diseño de esta etapa.

Ejemplo, sea el texto ”

Se debe transformar la frase anterior para que quede con la siguiente estructura:

	X_1	X_2	X_3	X_4	X_5	X_6	Y
Input 1	0	0	0	0	0	0	El
Input 2	0	0	0	0	0	El	gato
Input 3	0	0	0	0	El	gato	está
Input 4	0	0	0	El	gato	está	arriba
Input 5	0	0	El	gato	está	arriba	de
Input 6	0	El	gato	está	arriba	de	la
Input 7	El	gato	está	arriba	de	la	mesa

Tabla 24 Ejemplo datos de entrada LSTM

Sea una red LSTM de una capa, el caso descrito arriba se ve de la siguiente manera:

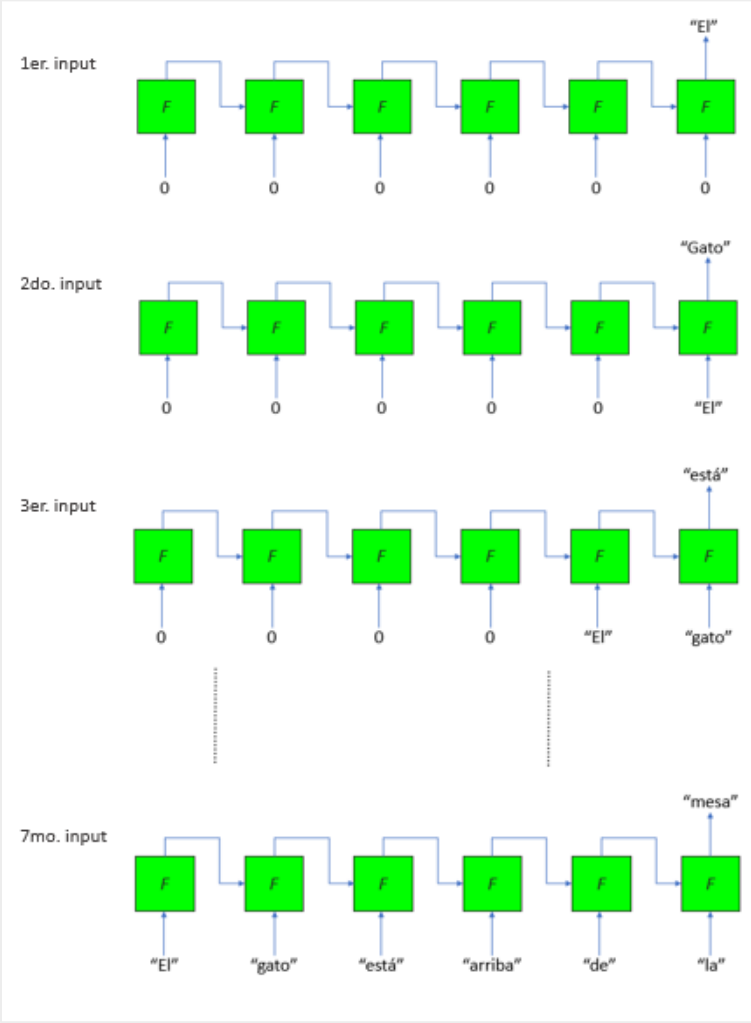


Figura 22 Ejemplo LSTM

De acuerdo con lo visto en la figura, se tiene una red LSTM para procesar una oración se debe definir el tamaño de la secuencia, que en este caso es de 6.

Para transformar cada oración hacia el arreglo requerido, se utiliza la función *pad_sequences* que trae la librería Keras.

En esta función, se debe especificar el tamaño de la secuencia. En nuestro caso hemos fijado este valor en 5.

Hay que recordar que, del proceso de limpieza y procesamiento del texto, hemos quedado con 1.855.105 oraciones. En esta etapa he comenzado a presentar problemas en cuanto a los requerimientos de memoria de mi computadora.

Al comenzar a realizar el padding de las oraciones, de las 1.855.105, sólo llega a realizar el padding de 800.000 de las oraciones, gatillando un error de memoria, generando así una lista de 14.510.254 filas. (que tienen la forma de la tabla 24, solo que esta es de 6 columnas).

De todos modos, al realizar lo anterior, el equipo queda con sobre un 80% de Memoria utilizada, por lo que es prácticamente inviable realizar otra operación.

Por otro lado, como se ha visto en los antecedentes de LSTM, en la salida de la red, se tiene un vector ***y*** del tamaño del vocabulario. Este vector **y** tendrá la distribución de probabilidad de cada palabra para una entrada específica en la red.

En Keras, es necesario que se defina el vector ***y*** de salida en forma de un vector con codificación one-hot.

El problema que esto genera es que, dadas las 1.855.105 oraciones, se obtiene un vocabulario de 569.704 palabras, obtenido a través de la función *tokenizer* de Keras aplicado al texto.

Las dimensiones del vector ***y***, por un lado, corresponde al tamaño del vocabulario, y por otro lado corresponde al tamaño del arreglo luego del padding, que vimos que, para 800.000 oraciones, se genera un arreglo de 14.510.254 filas.

Por lo tanto, para ese caso, el vector ***y*** es de tamaño 14.510.254 x 569.704.

Dado lo anterior, ejecutar LSTM con el total de oraciones en mi computador ha sido imposible.

En vista de esto, se ha optado por ejecutar esta etapa en Google Colab, que es un entorno gratuito de Jupyter Notebook que no requiere configuración y que se ejecuta completamente en la nube, con la posibilidad de usar GPU para la ejecución de los scripts.

Luego de cargar los datos en Google Drive, para importarlos desde Google Colab, se comienza a probar iterativamente distintos tamaños de muestra para el corpus de entrada, para ver cuál es la mayor cantidad de datos posible con los que se puede entrenar la red.

De lo anterior, he obtenido que, del total de 1.855.105 oraciones disponibles en nuestro corpus, solo se pueden usar 6.000 por motivos de cómputo.

Sobrepasando las 6.000 oraciones, Google Colab se desconecta por no disponibilidad de memoria.

De esta manera, usando solo 6.000 oraciones, se obtiene un vocabulario de 18.114 palabras. El tamaño del arreglo después de realizar el padding es de tamaño 109.868, por lo tanto, el tamaño del vector de salida ***y*** es de 109.868 * 18.114.

A continuación, se define la red a utilizar. Para esta implementación, dado que se está utilizando Keras, se requiere crear una red secuencial.

Creación de estructura LSTM en Keras

En esta etapa, se utilizará la forma secuencial de construir redes de aprendizaje profundo. Esta metodología permite apilar capas en la red de forma sencilla.

La red a implementar en este caso es la siguiente:

```
model = Sequential()
model.add(Embedding(vocab_size, hidden_size, input_length=num_steps))
model.add(LSTM(hidden_size))
model.add(Dense(vocab_size, activation='softmax'))
```

Figura 23 Red secuencial para implementar LSTM en Keras.

El primer paso consiste en crear un modelo de Keras con el constructor Sequential().

La primera capa en la red, según el diagrama de arquitectura mostrado anteriormente, es una capa de Embedding. Esto convertirá las palabras en vectores. A esta capa de Embedding() hay que ingresarle el tamaño del vocabulario como su primer argumento, luego la dimensión que se desea para el embedding, que en nuestro caso se ha escogido 100, para evitar problemas de desfalco de memoria. Finalmente, como esta es la primera capa de la red, se debe especificar la longitud de la secuencia de entrada, que en nuestro caso corresponde a 5.

La segunda capa es la correspondiente a LSTM. Para especificar una capa de LSTM, primero se tiene que proporcionar el número de nodos en las capas ocultas dentro de la celda LSTM, por ejemplo, el número de celdas en la capa de puerta de olvido, la capa de entrada de aplastamiento de tanh, etc.

La capa de salida está compuesta por una neurona para cada palabra del vocabulario y utiliza una función de activación de softmax para asegurar que la salida se normalice para que parezca una probabilidad. Es decir, la suma de todas las salidas en un momento debe ser igual a 1.

De este modo, la arquitectura de la red queda de la siguiente manera:

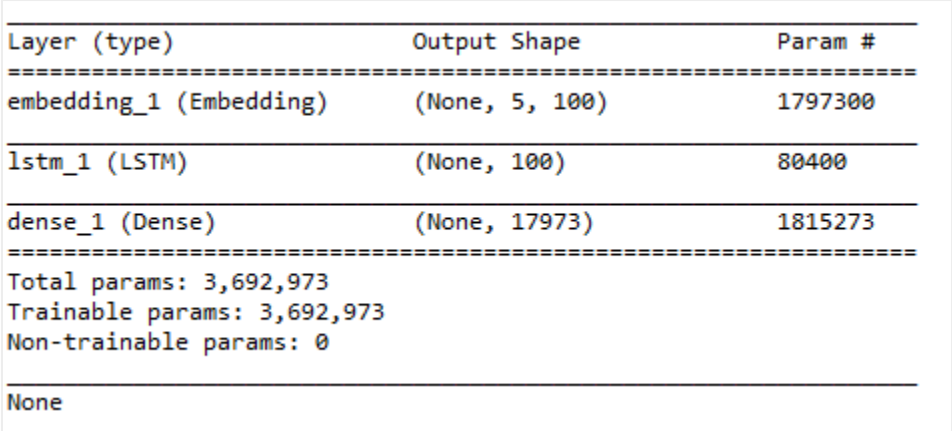


Figura 24 Sumario del modelo secuencial implementado.

Para este modelo secuencial, se deben calcular 3.692.973

El siguiente paso en Keras, una vez completado el modelo, es ejecutar el comando de compilación.

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figura 25 Comando para compilar el modelo.

En este comando, se debe especificar el tipo de pérdida que Keras debe usar para entrenar al modelo. En este caso, estamos usando 'categorical_crossentropy' que es la entropía cruzada aplicada en casos donde hay muchas clases o categorías, de las cuales sólo una es verdadera.

Luego, el optimizador que se utilizará es el denominado “Adam”, que corresponde a un optimizador "general" con pasos adaptables. Finalmente, se especifica una métrica 'categorical_accuracy', que permite ver cómo mejora la precisión durante el entrenamiento.

Una vez definidos los parámetros del compilador y ejecutado ya se puede correr el modelo con los datos.

```
model.fit(X, y, epochs=200, verbose=2)
```

Resultados obtenidos.

El modelo, que ha demorado más de 8 horas en ejecutar las 200 iteraciones ha llegado a tener un accuracy de 88% en el corpus de entrenamiento.

Sin embargo, para evaluar si tiene un buen funcionamiento, para nuestro objetivo, realizaremos pruebas en donde debe evaluar que frase es mejor que otra, en función del puntaje obtenido o la probabilidad de ocurrencia entregada.

Prueba 1

Se utilizará la frase: "*Para que no caiga en malas manos*" primero en orden y luego en desorden.

Orden: El resultado que retorna el modelo de lenguaje para la frase en orden es:

```
P(w=que|h=para)=0.0020511625334620476
P(w=no|h=para que)=5.309297193889506e-05
P(w=caiga|h=para que no)=2.6051836671037118e-17
P(w=en|h=para que no caiga)=4.487915794015862e-05
P(w=malas|h=para que no caiga en)=6.428014028583796e-15
P(w=manos|h=para que no caiga en malas)=1.1767503638182575e-10
Prob. oración: 9.63121262076557e-53
```

Figura 26 Resultado modelo de lenguaje frase 1 ordenada

Desorden: El resultado que retorna el modelo de lenguaje para la frase en desorden *"Caiga que no malas para en manos "* es:

```
P(w=que|h=caiga)=0.001108940108679235
P(w=no|h=caiga que)=4.2339065586816105e-09
P(w=malas|h=caiga que no)=7.358843894619302e-22
P(w=para|h=caiga que no malas)=1.9045080183049956e-12
P(w=en|h=caiga que no malas para)=1.378007681296367e-07
P(w=manos|h=caiga que no malas para en)=9.687430055661523e-16
Prob. oración: 8.784197111248324e-67
```

Figura 27 Resultado modelo de lenguaje frase 1 desordenada

Vemos que, para la frase con estructura gramatical correcta, la probabilidad de la oración es de 9.6e-53, mayor a la de la frase desordenada 8.7e-67

En este caso el modelo de lenguaje a evaluado mejor a la frase ordenada.

Prueba 2

Se utilizará la frase: "*Buen día*" primero en orden y luego en desorden.

Orden: el resultado que retorna el modelo de lenguaje para la frase en orden es:

```
P(w=día|h=buen)=3.375907181180082e-05
Prob. oración: 3.375905813688827e-05
```

Figura 28 Resultado modelo de lenguaje frase 2 ordenada

Desorden: El resultado que retorna el modelo de lenguaje para la frase en desorden *"Caiga que no malas para en manos "* es:

```
P(w=buen|h=día)=4.193515223960276e-07
Prob. oración: 4.1935165061967296e-07
```

Figura 29 Resultado modelo de lenguaje frase 2 desordenada

Vemos que nuevamente para la frase con estructura gramatical correcta, la probabilidad de la oración, que es de 3.37e-05, es mayor a la de la frase desordenada 4.19e-07

En este caso el modelo de lenguaje a evaluado mejor a la frase ordenada.

Prueba 3

Se utilizará la frase: "*Los detalles nos hacen grandes*" primero en orden y luego en desorden.

Orden: el resultado que retorna el modelo de lenguaje para la frase en orden es:

```
P(w=detalles|h=los)=1.6555365789372445e-07
P(w=nos|h=los detalles)=6.633631578073507e-12
P(w=hacen|h=los detalles nos)=4.9025436894790594e-14
P(w=grandes|h=los detalles nos hacen)=1.3253298902782262e-07
Prob. oración: 7.135683924854742e-39
```

Figura 30 Resultado modelo de lenguaje frase 3 ordenada

Desorden: El resultado que retorna el modelo de lenguaje para la frase en desorden *" Los grandes nos hacen detalles"* es:

```
P(w=grandes|h=los)=8.181296834663954e-06
P(w=nos|h=los grandes)=1.3808881815791096e-10
P(w=hacen|h=los grandes nos)=1.017856284012738e-14
P(w=detalles|h=los grandes nos hacen)=5.305972849364535e-32
Prob. oración: 6.1014094208401454e-61
```

Figura 31 Resultado modelo de lenguaje frase 3 desordenada

Vemos que, para la frase con estructura gramatical correcta, la probabilidad de la oración, que es de 7.13e-39, es mayor a la de la frase desordenada 6.1e-61.

En este caso el modelo de lenguaje a evaluado mejor a la frase ordenada.

Prueba 4

Se utilizará la frase: “ primero en orden y luego en desorden.

Orden: el resultado que retorna el modelo de lenguaje para la frase en orden es:

```
P(w=he|h=nunca)=1.9775680470957013e-07
P(w=tenido|h=nunca he)=2.69852076204306e-15
P(w=tanto|h=nunca he tenido)=6.9440408284604516e-12
P(w=apoyo|h=nunca he tenido tanto)=7.758385578426896e-08
P(w=gracias|h=nunca he tenido tanto apoyo)=2.819506562445895e-06
P(w=por|h=nunca he tenido tanto apoyo gracias)=8.795713289655449e-12
P(w=todos|h=nunca he tenido tanto apoyo gracias por)=2.0831143013102917e-20
P(w=los|h=nunca he tenido tanto apoyo gracias por todos)=1.0534051929356192e-11
P(w=detalles|h=nunca he tenido tanto apoyo gracias por todos los)=4.645550871068549e-12
Prob. oración: 7.268259580333684e-99
```

Figura 32 Resultado modelo de lenguaje frase 4 ordenada

Desorden: El resultado que retorna el modelo de lenguaje para la frase en desorden *"Nunca he tenido todos apoyo, gracias los tanto por detalles"* es:

```
P(w=he|h=nunca)=1.9775680470957013e-07
P(w=tenido|h=nunca he)=2.69852076204306e-15
P(w=todos|h=nunca he tenido)=2.007297007367015e-05
P(w=apoyo|h=nunca he tenido todos)=1.1527494236940328e-11
P(w=gracias|h=nunca he tenido todos apoyo)=7.694313239881012e-07
P(w=los|h=nunca he tenido todos apoyo gracias)=1.60091818274255e-11
P(w=tanto|h=nunca he tenido todos apoyo gracias los)=9.314792860648513e-12
P(w=por|h=nunca he tenido todos apoyo gracias los tanto)=3.0907879033925667e-10
P(w=detalles|h=nunca he tenido todos apoyo gracias los tanto por)=2.2520507594187e-12
Prob. oración: 9.861945816518538e-87
```

Figura 33 Resultado modelo de lenguaje frase 4 desordenada

Vemos que, para la frase con estructura gramatical correcta, la probabilidad de la oración es de 7.26e-99, menor a la de la frase desordenada 9.86e-87.

En este caso el modelo de lenguaje a evaluado mejor a la frase desordenada.

Prueba 5

Se utilizará la frase: "*El uso de redes sociales y su mensajería privada podría ser un leve un indicio de esta opción* en orden y luego en desorden.

Orden: el resultado que retorna el modelo de lenguaje para la frase en orden es:

```
P(w=uso|h=el)=2.260433575429488e-06
P(w=de|h=el uso)=3.2931666282820515e-06
P(w=redes|h=el uso de)=4.3032694939531977e-17
P(w=sociales|h=el uso de redes)=2.1153191543634713e-27
P(w=y|h=el uso de redes sociales)=1.7520202391096973e-06
P(w=su|h=el uso de redes sociales y)=2.1137931682790878e-10
P(w=mensajería|h=el uso de redes sociales y su)=1.3732453204328712e-30
P(w=privada|h=el uso de redes sociales y su mensajería)=1.7138508900905666e-14
P(w=podría|h=el uso de redes sociales y su mensajería privada)=1.9616310664586148e-14
P(w=ser|h=el uso de redes sociales y su mensajería privada podría)=0.00010890130215557292
P(w=un|h=el uso de redes sociales y su mensajería privada podría ser)=1.0808244965687663e-08
P(w=leve|h=el uso de redes sociales y su mensajería privada podría ser un)=4.356571353669736e-11
P(w=un|h=el uso de redes sociales y su mensajería privada podría ser un leve)=5.578678528557954e-14
P(w=indicio|h=el uso de redes sociales y su mensajería privada podría ser un leve un)=9.06509405301098e-13
P(w=de|h=el uso de redes sociales y su mensajería privada podría ser un leve un indicio)=1.9770183712353173e-08
P(w=esta|h=el uso de redes sociales y su mensajería privada podría ser un leve un indicio de)=8.429257861370179e-14
P(w=opción|h=el uso de redes sociales y su mensajería privada podría ser un leve un indicio de esta)=4.788545084011275e-06
Prob. oración: 2.397513924304599e-201
```

Figura 34 Resultado modelo de lenguaje frase 5 ordenada

Desorden: El resultado que retorna el modelo de lenguaje para la frase en desorden *"El uso de sociales redes y su privada mensajería podría ser un leve un indicio de esta opción"* es:

```
P(w=uso|h=el)=2.260433575429488e-06
P(w=de|h=el uso)=3.2931666282820515e-06
P(w=sociales|h=el uso de)=6.934596967766993e-06
P(w=redes|h=el uso de sociales)=4.373522798002725e-23
P(w=y|h=el uso de sociales redes)=1.306836860948124e-10
P(w=su|h=el uso de sociales redes y)=4.7574841843811555e-09
P(w=privada|h=el uso de sociales redes y su)=2.2185777853584878e-32
P(w=mensajería|h=el uso de sociales redes y su privada)=6.908534549011 74e-15
P(w=podría|h=el uso de sociales redes y su privada mensajería)=1.2047062676387031e-15
P(w=ser|h=el uso de sociales redes y su privada mensajería podría)=2.8500417101895437e-05
P(w=un|h=el uso de sociales redes y su privada mensajería podría ser)=2.007637704082299e-06
P(w=leve|h=el uso de sociales redes y su privada mensajería podría ser un)=1.4325944999272827e-16
P(w=un|h=el uso de sociales redes y su privada mensajería podría ser un leve)=5.1429472253160594e-14
P(w=indicio|h=el uso de sociales redes y su privada mensajería podría ser un leve un)=9.06509405301098e-13
P(w=de|h=el uso de sociales redes y su privada mensajería podría ser un leve un indicio)=1.9770183712353173e-08
P(w=esta|h=el uso de sociales redes y su privada mensajería podría ser un leve un indicio de)=8.429257861370179e-14
P(w=opción|h=el uso de sociales redes y su privada mensajería podría ser un leve un indicio de esta)=4.788545084011275e-06
Prob. oración: 7.904037337366051e-196
```

Figura 35 Resultado modelo de lenguaje frase 5 desordenada

Vemos que, para la frase con estructura gramatical correcta, la probabilidad de la oración es de 2.3e-201, menor a la de la frase desordenada 7.9e-196

En este caso el modelo de lenguaje a evaluado mejor a la frase desordenada.

Es muy difícil que un modelo de lenguaje creado en base a tan poco texto, solo 6000 oraciones, tenga un buen rendimiento. En el caso de las pruebas, al parecer las oraciones cortas y con palabras comunes pueden tener mejor rendimiento, ya que es más posible que estén dentro de esas 6000 oraciones con que se entrenó el modelo.

La dificultad de crear un buen modelo de lenguaje radica en el costo computacional que requiere. Si tan sólo usando 6000 oraciones, una dimensionalidad del embedding de 100 y 5 pasos de secuencia, el modelo ha tomado alrededor de 8 horas en entrenar, pensar en entrenar un modelo de millones de oraciones es sólo posible si se cuenta con múltiples GPU y bastante memoria disponible.

Para finalizar el proyecto, se debe juntar todo lo realizado hasta ahora, es decir, las traducciones por palabras realizadas en base al Word embedding, para luego poder traducir los chilenismos presentes en las frases que se desean traducir y luego poder evaluar las propuestas de traducción en este modelo de lenguaje.

En la siguiente sección se comenta esta última parte.

5.6 Traducción de frases desde el chileno al español

La finalidad de este proyecto es obtener un sistema que permita obtener una traducción de frases que contengan chilenismos al español, manteniendo el sentido y estructura.

Dado que no se han encontrado otros proyectos con traducciones del chileno al español, no es posible realizar benchmarking de performance versus otros sistemas. La evaluación que se puede realizar es usar ejemplos obtenidos de páginas web que tratan este problema de forma más coloquial.

Tal como se ha descrito en la etapa de diseño, el proceso de traducción comprende las siguientes partes:

- Ingresar oración en chileno a traducir, de largo **n** palabras, con **X** chilenismos
- Separar la oración en palabras y obtener la traducción de cada chilenismo por separado. Se obtendrán **Y** traducciones por chilenismo, con **Y** a definir.

- Se generan $\mathbf{Y^x}$ oraciones a partir de la combinatoria de las posibles traducciones.
- Se evalúa cada una de las oraciones obtenidas en el modelo de lenguaje y se retornan aquellas que tengan el mejor resultado evaluado en el modelo.

A continuación, se realizan pruebas siguiendo todos los pasos, para que describir la metodología:

Prueba 1

Frase que traducir: ***ó***

El proceso por seguir para obtener la traducción es el siguiente:

1. Se obtienen 3 traducciones del chilenismo presente en la frase. En este caso, “cachai”.

Chilenismo	**Cachai**
Traducción al español	demuestra
	sabes
	sabe

2. Se generan las frases obtenidas por las traducciones:
   - *Demuestra dónde queda esta calle*
   - *Sabes dónde queda esta calle*
   - *Sabe dónde queda esta calle*

3. Se evalúan las frases en el modelo de lenguaje:
   - *P(Demuestra donde queda esta calle) = 2.9781221261467616e-57*
   - *P(Sabes dónde queda esta calle) = 1.2531690393328837e-45*
   - *P(Sabe dónde queda esta calle) = 1.591714589706135e-45*

4. La traducción escogida es aquella que presenta el mayor puntaje en el modelo de lenguaje. En este caso, las dos últimas frases tienen valores muy parecidos, y coincide con que ambas traducciones serían una correcta traducción de la frase original.

Cachai dónde queda esta calle -> Sabe donde queda esta calle

Prueba 2

Se usará un tweet real escrito en Chile.

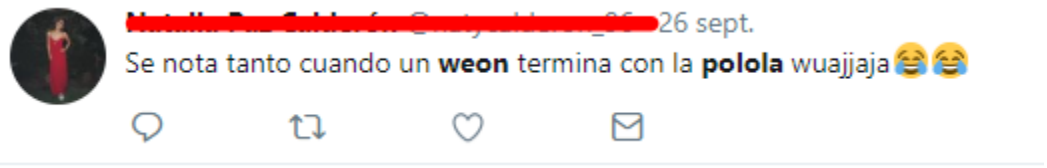


Figura 36 Tweet chileno usado como ejemplo para realizar traducción.

Frase: “Se nota tanto cuando un weon termina con la polola wuajaja”

1. Se obtienen 3 traducciones de los chilenismos presentes en la frase. En este caso, “weon” y “polola”

Chilenismo	weon	polola
Traducción al español	Tio	Amiga
	Tío	Novia
	chico	hermana

2. Se generan las frases obtenidas por las traducciones: Y=3, X=2 => **$Y^x = 9$**
   - Se nota tanto cuando un tio termina con la amiga
   - Se nota tanto cuando un tio termina con la novia
   - Se nota tanto cuando un tio termina con la hermana
   - Se nota tanto cuando un tío termina con la amiga
   - Se nota tanto cuando un tío termina con la novia

- Se nota tanto cuando un tío termina con la hermana
- Se nota tanto cuando un chico termina con la amiga
- Se nota tanto cuando un chico termina con la novia
- Se nota tanto cuando un chico termina con la hermana
- Se nota tanto cuando un weon termina con la wuajaja

3. Se evalúan las frases en el modelo de lenguaje:
   - Se nota tanto cuando un tio termina con la amiga= 2.3549439677e-88
   - Se nota tanto cuando un tio termina con la novia = 9.6298281873e-84
   - Se nota tanto cuando un tio termina con la hermana = 1.17788105e-81
   - Se nota tanto cuando un tío termina con la amiga= 2.5224313885e-92
   - Se nota tanto cuando un tío termina con la novia = 6.294519412992e-87
   - Se nota tanto cuando un tío termina con la hermana= 3.08502947605e-88
   - Se nota tanto cuando un chico termina con la amiga= 4.72294770804e-100
   - Se nota tanto cuando un chico termina con la novia=1.2118661150207e-94
   - Se nota tanto cuando un chico termina con la hermana=5.5025794796e-100

4. La traducción escogida es aquella que presenta el mayor puntaje en el modelo de lenguaje. En este caso, corresponde a

   “Se nota tanto cuando un tio termina con la hermana”

De lo visto en las traducciones anteriores, polola se refiere a novia, por lo que está traducción no tiene mucho sentido.

En realidad, considero que hacer más pruebas no tiene sentido debido al mal modelo de lenguaje que se ha obtenido. No es posible que pueda generalizar con la cantidad tan pequeña de texto con el que se ha podido modelar.

En distintos sitios se comenta que un modelo de lenguaje debe tener desde 100.000 palabras hacia arriba en su vocabulario. En este caso, el vocabulario contiene solo

17.000 palabras, y con sólo 6.000 textos es imposible que pueda tener una visión amplia del lenguaje.

En resumen, si se tiene un modelo de lenguaje malo, no se puede confiar en el resultado que éste otorga. Por lo tanto, es imperioso para este tipo de labores contar con una capacidad de computo importante, para no sufrir este tipo de percances.

6 CONCLUSIONES Y LÍNEAS DE FUTURO

6.1 Conclusiones

El desarrollo de este proyecto a sido complejo, ya que consta de varias etapas, donde cada una de ellas implico programar y aprender temas nuevos.

Ha sido realmente interesante ver el estado actual de los sistemas de traducción automática. Varias de las compañías grandes están invirtiendo esfuerzo en ello, como Google, Facebook, etc. y desarrollando nuevas técnicas para lograr que mediante aprendizaje no supervisado se puedan obtener traductores capaces de obtener mejores rendimientos que los actuales, y no depender de corpus paralelos, que muchas veces no existen.

Sobre los resultados obtenidos, queda una sensación amarga al no haber podido obtener un buen modelo de lenguaje que sirviese para evaluar las frases candidatas. Sin embargo, queda la esperanza de que la metodología tiene mucho por mejorar y eventualmente, ofrecer buenas traducciones.

El problema de no contar con recursos computacionales de alta demanda limita el uso de herramientas tan poderosas como son las de Deep learning. Redes recurrentes bidireccionales pueden aportar mucho a crear modelos que sean capaces de representar los efectos gramaticales y sintácticos del lenguaje.

Queda también la visión de que este es un sistema que tiene la capacidad de realizar buenas traducciones, si se desarrollara con una gran cantidad de texto, hablando sobre los 100 millones de frases, y con la capacidad de computo requerida para efectuar un buen modelo de lenguaje. Sería una plataforma que podría ofrecerse a usuarios para que puedan realizar sus consultas online.

En general, a modo personal, ha sido un trabajo intenso. Cada una de las etapas me requirió tiempo tanto de estudio como de desarrollo.

Quedo sumamente satisfecho con lo aprendido, pero con una espina respecto al resultado final.

A continuación, se comentan algunas conclusiones respecto a cada objetivo específico.

6.2 Conclusiones por objetivos específicos

Extracción y limpieza de textos para formar corpus.

Me ha sorprendido la cantidad de información y herramientas que existen para realizar esta labor. Este trabajo se ha llevado a cabo solo usando texto extraído desde internet.

He podido adaptarme a cada tipo de fuente de la que tenía que extraer información y, siendo que no tengo mayores conocimientos de programación, ha sido relativamente fácil poder adecuar los scripts a cada fuente.

Lo que más me ha gustado de esta etapa, es que, si quiero, por ejemplo, probar algoritmos para análisis de sentimientos o algún otro proyecto, puedo hacerlo con los textos que a mí me interesen, ya que podría buscar donde se encuentran y extraerlos con estas herramientas.

Por otro lado, el uso de expresiones regulares para la limpieza de texto facilita mucho esta labor. Aprender esto es algo que requiere de un tiempo de dedicación ya que tiene detalles que lo hacen una herramienta muy poderosa a la hora de trabajar con textos.

Sin duda, que conocer este tipo de herramientas es fundamental para poder recolectar datos para futuros proyectos.

Word Embedding

A decir verdad, me ha impresionado el resultado de este algoritmo, considerando lo sencillo que es, una red neuronal de una capa oculta.

Creo que, para realizar labores de procesamiento natural de lenguaje, como clasificación de textos, análisis de sentimientos, etiquetado de entidades, etc., esta herramienta va a ser lo primero que utilice.

Sin duda, su rendimiento puede mejorar utilizando un corpus de mayor tamaño, pero también existen los requerimientos computacionales que limitan la cantidad de texto que se puede procesar sin GPU, como ha sido mi caso.

Otro punto que me ha parecido muy interesante es el caso de Skip-gram, el cual, puede acomodar o representar palabras que suelen ir juntas, de manera cercana, como se vio con el ejemplo de “comer” y que Skip-gram ha devuelto la palabra “carne”. Con esto se podría obtener un sistema para ver cuales adjetivos acompañan, por ejemplo, a instituciones públicas para ver cuál es la percepción de las personas frente a ellas.

No me cabe duda de que los métodos aprendidos de Word embedding son una herramienta que volveré a utilizar en cualquier proyecto de PNL que se me presente.

Traslación de un espacio de vectores a otro

Esta etapa ha sido una gran sorpresa, ya que me los resultados obtenidos para las traducciones de chilenismos creo que han sido bastante correctos.

Si bien, se pueden evaluar distintas formas de realizar el mapeo, a diferencia de lo realizado aquí, la solución propuesta creo que ha cumplido su objetivo.

Cabe recordad que este mapeo se suele hacer usando un diccionario bilingüe. La solución propuesta de usar las palabras que se encuentran en ambos corpus ha parecido adecuada. Por motivos de tiempo no se he podido evaluar si quizás algún subconjunto de ese diccionario podría haber funcionado mejor, en vez de usar todas las 57.000 palabras.

Creo que el desafío es poder realizar este mapeo entre idiomas distintos y sin contar con un diccionario bilingüe. Considerando que los distintos idiomas presentan morfologías parecidas en la distribución espacial, encontrar algún modelo que se base en la forma en que se encuentran distribuidas las palabras y encuentre alguna manera de efectuar las rotaciones y traslaciones necesarias sería muy interesante.

Modelo de lenguaje

Sin duda, que realizar un modelo de lenguaje requiere de muchísima capacidad de cómputo. Me sabe mal no haber podido usar ni siquiera un 1% del total de texto que tenía para obtener el modelo, por no tener suficiente recurso computacional.

Creo que este es el punto débil del proyecto, ya que he obtenido un modelo de lenguaje muy básico, entrenado con muy poco texto. Sin duda que es algo que queda pendiente de mejorar en la medida que se cuenta con mejor capacidad de cómputo.

De todos modos, me ha parecido muy interesante todos los usos que se le pueden dar a un modelo de lenguaje, desde generación de texto basado en texto de cierto dominio, hasta evaluación de oraciones tanto hablado como escrito, reconocimiento del habla, etc.

Sistema de traducción

La solución que he propuesto, si bien, funciona, al depender tanto del modelo de lenguaje, puede terminar devolviendo traducciones que no son correctas gramaticalmente.

Este proyecto tiene 3 factores que son clave: realizar una correcta representación de las palabras mediante el Word embedding. Realizar un buen mapeo entre los espacios y obtener un modelo de lenguaje basado en un corpus de gran volumen.

Al fallar en el modelo de lenguaje, el proyecto queda un tanto cojo. Pero es algo que se debe mejorar a futuro, ahora que ya conozco cuales son dificultades de realizar algo así.

Por otro lado, creo que obtener un número de frases basado en la combinación de las **X** traducciones puede ser algo lento para un traductor en tiempo real. Queda pendiente evaluar que otra forma se puede realizar para acelerar ese proceso y no tener que evaluar 1000 frases en un modelo para obtener cual es la correcta.

Finalmente, queda mucho por hacer en este proyecto, tal como se comenta en la siguiente sección.

6.3 Líneas de futuro.

Sin duda que hay bastantes cosas que se pueden realizar y que han quedado pendientes ya sea por tiempo y/o por recursos computacionales.

A continuación, enumero algunas de las cosas que desearía realizar a futuro con este proyecto.

- Extraer más texto. Creo que ha sido una buena decisión obtener texto desde distintas fuentes, donde se mezclan las formas de expresarse, para capturar de forma más integral las características del lenguaje. Me gustaría poder trabajar con al menos 10 millones de oraciones por lengua, para ya poder tener una representación más generalizada.

- En el futuro, al momento de tokenizar las oraciones, es decir, pasarlas a palabras, se debería considerar aquellas expresiones que están compuestas por dos o más palabras, como, por ejemplo: “por supuesto”, “muchas gracias”, “buen día”. Al igual que nombres como “Real Madrid”, “Universidad Ramón Llull”, etc. Esto sin duda que le agrega riqueza semántica a la representación vectorial que se obtenga.

- Además, al momento de limpiar el texto, sería de mucho valor poder corregir las faltas de ortografía. Esto con el fin de disminuir el tamaño del vocabulario, y de ofrecer traducciones de palabras escritas correctamente.

- Debido a que cada vez que se entrena el modelo de Word embedding toma al menos 6 - 8 horas, en un futuro, se debiese hacer un benchmarking más exhaustivo de los parámetros, por ejemplo, la ventana de contexto, la cantidad mínima de palabras, etc. Por motivos de tiempo, solo se han realizado un par de pruebas y el esquema usado fue escogido porque fue el que obtuvo mejores resultados.

- Tal como se comentó antes, evaluar si existe algún subconjunto del diccionario obtenido por la intersección de ambos corpus, para realizar la regresión, tendría un mejor resultado. Quizás considerar solo las palabras más frecuentes, o solo subconjunto de palabras que tengan una representación geométrica similar por sobre algún umbral, etc.

- Para el modelo de lenguaje, asumiendo que se tiene una capacidad de cómputo suficiente para poder entrenar con todo el texto disponible, sería interesante utilizar LSTM bidireccional. De esta forma, para la evaluación del texto no sólo importa lo que está antes del texto, sino que también lo que está adelante, siendo un modelo mucho más efectivo, pero mucho más caro computacionalmente.

- Tal como se comentó antes, optimizar la generación de frases candidatas de traducción.

7 REFERENCIAS

Alexa. (2018). *Alexa Top Sites in Chile*. Retrieved from Alexa: https://www.alexa.com/topsites/countries/CL

Bird, S., Klein, E., Loper, E., Garrette, D., Ljunglof, P., Nothman, J., . . . Dimitriadis, A. (s.f.). *The nltk project*. Obtenido de http://www.nltk.org

Cardellino, C. A. (2015). *Representación de Palabras Mediante Vectores.* Obtenido de http://www.famaf.proed.unc.edu.ar/pluginfile.php/20437/mod_resource/content/2/PF-Cardellino.pdf

Gensim. (2018). *Gensim, Topic Modelling for humans*. Obtenido de Gensim: https://radimrehurek.com/gensim/

Henrique, J. (2018). *GetOldTweets-python*. Obtenido de https://github.com/Jefferson-Henrique/GetOldTweets-python

Hochreiter, S., & Schmidhuber, J. (1997). LONG SHORT-TERM MEMORY. *Neural Computation*. Obtenido de LONG SHORT-TERM MEMORY

Lavinia, P. (2018). *Manual para entender el español de un chileno*. Obtenido de Intriper: https://intriper.com/manual-para-entender-el-castellano-chileno/

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space.* Obtenido de https://arxiv.org/abs/1301.3781

Mikolov, T., V. Lee, Q., & Sutskever, I. (2013). *Exploiting Similarities among Languages for Machine Translation.*

Mikolov, Tomas; Quoc, Le; Sutskever, Ilya. (2013). *Exploiting Similarities among Languages for Machine Translation.* Obtenido de https://arxiv.org/abs/1309.4168

Numpy. (s.f.). Obtenido de http://www.numpy.org/

ŘEHŮŘEK, R., & SOJKA, P. (2010). Software Framework for Topic Modelling with Large Corpora. *Proceedings of LREC 2010 workshop New Challenges for NLP*

Frameworks (págs. 46-50). Valletta: University of Malta. Obtenido de https://is.muni.cz/publication/884893/en

Scikit-learn. (2018). *Scikit-learn*. Obtenido de Sklearn.linear_model.LinearRegression: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. Obtenido de http://www.aclweb.org/anthology/P10-1040

Wikipedia. (2018). *Anexo:Comunas de Chile por población*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Anexo:Comunas_de_Chile_por_poblaci%C3%B3n

Wikipedia. (2018). *Anexo:Provincias y ciudades autónomas de España*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Anexo:Provincias_y_ciudades_aut%C3%B3nomas_de_Espa%C3%B1a

Wikipedia. (2018). *Beautiful Soup*. Obtenido de https://es.wikipedia.org/wiki/Beautiful_Soup

Wikipedia. (2018). *Word embedding*. Obtenido de https://es.wikipedia.org/wiki/Word_embedding