

**TAREA no. 2**

*Inteligencia Artificial (TICS315)*

*Tópico: Procesamiento de Lenguaje Natural*

Prof. John Atkinson

**Publicación:** Miércoles 30 de Agosto de 2022

**Entrega:** Lunes 10 de Octubre de 2022

**OBJETIVO**

El objetivo de esta tarea es entender y aplicar conceptos básicos de *Word Embeddings* para abordar una actividad simple de NLP tal como la generación automática de resúmenes (*Summarization)*.

**DESCRIPCIÓN**

Muchas decisiones que se toman en diversos ámbitos productivos dependen de la síntesis de mucha documentación que se encuentra en la Web. Sin embargo, la cantidad de información disponible es tal que es prácticamente imposible resumir toda la información en forma manual. El problema es que sin una síntesis o resumen, puede tomar mucho tiempo entender algún documento, artículo o reporte, de ahí es que se requiere de métodos automatizados. El generador más simple de resúmenes (aka. *Summarizer*) es un algoritmo que determina la importancia de las oraciones del texto, y luego extrae aquellas que son más importantes y que son coherentes entre ellas, produciendo el resumen final.

Una forma de abordar esta actividad de *summarization* consiste en utilizar modelos de *word embeddings* tales como ***Word2Vec****.* Luego, se puede generar las representaciones semánticas (embeddings) de cada oración de un texto completo. Con dichos embeddings se puede determinar la importancia de cada oración en el texto, y luego extraer las más importantes para ir agregándolas al resumen final.

Con el fin de que el texto resumido final sea coherente, el objetivo es agregar oraciones que son importantes pero que a la vez tienen una alta similitud con las que estaban hasta ahora en el resumen. Este último paso asegura que las oraciones que se producen en el resumen estén conectadas *lógicamente* con las oraciones previas. Esta característica lingüística de los textos comprensibles se denomina *coherencia local*.

Un algoritmo simple tentativo que permite producir un resumen de N oraciones en base a un texto completo utilizando el enfoque anterior, es el siguiente:

**GENERAR\_RESUMEN**(O, N, P, U)

**Input**: **O** (Oraciones del texto completo), **P** (Puntajes de cada oración)

**N** (número de oraciones del resumen), **U** (Umbral de similitud)

**Output**: Resumen

O 🡨 **OrdenarOraciones**(O,P)

Largo 🡨 1

m 🡨 **Largo**(O)

Resumen 🡨 {}

**for** i 🡨 1 **to m** **do**

**If** (Largo > **N**) **then** **return** Resumen

Vo 🡨 **ObtenerVector**(O[i])

incluirOracion 🡨 False

**for** j 🡨 1 **to** Largo **do**

Vr 🡨 **ObtenerVector**(Resumen[j])

Sim 🡨 **Similitud**(Vo,Vr)

**If**  Sim > U

**and** O[i] no incluida en Resumen **then**

incluirOracion 🡨 True

**If** incluirOracion **then**

Resumen[Largo] 🡨 O[i]

Largo 🡨 Largo + 1

Primero, las oraciones (**O**) se ordenan en orden descendiente de sus puntajes de “calidad” (**P**), lo que se lleva a cabo por la función **OrdenarOraciones(O,P)**. El puntaje de cada oración se calcula como el *coseno* entre su vector y el vector *centroide* del documento (i.e., el centroide es el promedio de los vectores de las oraciones del documento).

Las mejores oraciones se seleccionan iterativamente y se agregan al resumen hasta que se alcance el número máximo de oraciones a producir (**N**). Durante cada iteración, se calcula la similitud *coseno* entre la siguiente posible oración (**O[ i ]**) y las que ya están en el resumen (**Resumen[ j ]**). Se descarta la nueva oración si es que el valor de similitud no sobrepasa el umbral **U**.

La función **ObtenerVector(Oración)** simplemente entrega el vector (embedding) promedio de las palabras de **Oración**, y la función **Similitud(V1,V2)**, computa la similitud *coseno* entre vectores **V1** y **V2**.

Note que en caso que lo requiera, la biblioteca NLTK posee funciones para *tokenización* de palabras (word\_tokenize(..)) o de oraciones (sent\_tokenize(..)), que permiten separar una frase en palabras o un texto completo en oraciones, respectivamente. Los detalles se pueden obtener acá: <https://www.nltk.org/api/nltk.tokenize.html>.

El objetivo de la tarea es implementar el algoritmo **GENERAR\_RESUMEN(..)** y las funciones relacionadas, para generar un resumen a partir de un documento completo. Para ello, Ud. deberá primero **entrenar** un modelo de Word embedding con el método *Word2Vec* a partir de una muestra de casi 700 *discursos presidenciales* disponibles en *WebCursos*. Los vectores resultantes para cada palabra del corpus son los que utilizará el algoritmo de selección de oraciones previamente detallado.

**CONSIDERACIONES**

La tarea se debe realizar en grupos de **3 estudiantes**, y se deberá entregar lo siguiente:

1. Programas en Python (Colab) comentados que permitan entenderlos.
2. Mini-informe técnico de un **máximo** **de 2 páginas** que describa lo siguiente:
   * El enfoque de solución que se utilizó (ej. un diagrama de procesos o arquitectura que indique tareas de análisis de lenguaje utilizadas – **NO código**-, etc).
   * Problemas detectados con el método de summarization.
   * Breve descripción del rol de cada uno de los miembros del grupo.

Como ayuda, Ud. puede re-utilizar parte de los ejemplos en *Python* dados por el profesor en las clases.