UNIVERSIDAD
DE GRANADA

Sistemas Inteligentes para la Gestión en la Empresa

Procesamiento de lenguaje natural: *word embeddings*

Trabajo de teoría

Víctor Vázquez Rodríguez
victorvazrod@correo.ugr.es

Máster Universitario en Ingeniería Informática

Curso 2019/20

Índice

1. Introducción

Hoy en día, el procesamiento de lenguaje natural (NLP por sus siglas en inglés) es uno de los principales campos de estudio en inteligencia artificial y *deep learning*. Este campo se centra en la interpretación del lenguaje humano por parte de ordenadores, permitiendo cosas como el control de dispositivos con la voz o el análisis y traducción de textos.

Un aspecto muy importante a la hora de procesar y analizar lenguaje natural es la representación que usamos del mismo, concretamente, cómo representamos las palabras de forma que un ordenador pueda trabajar con ellas de forma eficiente y obtener información relevante de su significado y su contexto. Desde el punto de vista del *machine learning*, podríamos interpretar las palabras de un texto (o de cualquier conjunto de palabras) como valores de una variable categórica, donde cada palabra distinta supone una categoría. Uno de los métodos más comunes y usados para representar este tipo de variables es la codificación *one-hot*, que nos permite obtener un vector de valores númericos para cada categoría.

No obstante, esta técnica presenta grandes limitaciones para el procesamiento de lenguaje natural, de forma que surgen los *word embeddings*. Se trata de modelos apoyados en las redes neuronales para la proyección o incrustación (traducción literal de *embedding*) de palabras en un espacio vectorial [1]. En la sección 2 de este documento se explican las limitaciones de la codificación *one-hot* y por qué son necesarios los *word embeddings*, mientras que en la sección 3 se exponen algunas de las técnicas más utilizadas.

2. Limitaciones de la codificación *one-hot*

La codificación *one-hot* es un método muy sencillo para la representación de los distintos valores de una variable categrórica como vectores. Si tenemos N categorías diferentes, la codificación *one-hot* de una de ellas sería un vector de tamaño N con $N-1$ ceros y un uno en la posición cuyo índice representa la categoría. Para entender mejor su funcionamiento y, además, su aplicación a la representación de palabras, vamos a ver un ejemplo.

Consideramos las frases *Have a good day* y *Have a great day*. Si construimos el vocabulario de estas frases, obtenemos el siguiente conjunto:

$$V = \{Have, a, good, great, day\} \tag{1}$$

Aplicando la codificación *one-hot*, obtenemos las siguientes representaciones vectoriales para cada una de las palabras del vocabulario V:

- $Have = [1, 0, 0, 0, 0]$
- $a = [0, 1, 0, 0, 0]$
- $good = [0, 0, 1, 0, 0]$
- $great = [0, 0, 0, 1, 0]$
- $day = [0, 0, 0, 0, 1]$

Estos vectores forman parte de un espacio con 5 dimensiones, donde cada vector es la representación de una palabra en dicho espacio. Cuando nos fijamos en el ejemplo y en esta visualización, nos damos cuenta de las dos limitaciones principales de la codificación *one-hot* [2]:

- Para variables con una alta cardinalidad, como puede ser el caso del ejemplo per con un texto con muchas palabras, la dimensionalidad de los vectores se vuelve demasiado grande.
- Los vectores obtenidos no tienen proyecciones sobre ninguna otra dimensión excepto la suya, de forma que la distancia entre todos ellos es la misma.

En concreto, este último punto es especialmente problemático cuando lo que estamos representando son palabras. Volviendo al ejemplo, los vectores de *good* y *great* tienen la misma distancia entre ellos que *day* y *Have*, dando a entender que son igual de similares, lo cuál no es cierto. Con la codificación *one-hot* perdemos la información sobre la similitud entre palabras y su contexto. Los *word embeddings* surgen para solucionar esto, permitiendo obtener representaciones vectoriales de las palabras que reflejen en el espacio su relación con otras (significado, similitud). No se trata solo de representar las palabras como vectores, sino también de poder aplicar operaciones cartesianas sobre ellos para trabajar con el lenguaje de una forma relativamente sencilla para el ordenador.

Estos *embeddings* se suelen conseguir entrenando modelos, algunos de ellos basados en redes neuronales. En la siguiente sección se va a explicar el funcionamiento de los más comunes.

3. Principales técnicas de *word embedding*

Como ya se ha indicado, los *word embeddings* son modelos entrenados para la obtención de representaciones vectoriales a partir de palabras. Estos modelos pueden ser entrenados de forma individual para realizar su labor o como parte de una tarea concreta de NLP, obteniendo unas representaciones más adaptadas al problema específico [3]. De todas formas, un modelo ya entrenado se puede reutilizar para tareas de NLP diferentes, pudiendo actualizarse para adaptarse a las necesidades concretas de cada caso.

En las siguientes subsecciones se presentan y explican brevemente algunas de las técnicas más conocidas de *word embedding*.

3.1. Word2vec

3.1.1. *Continuous bag-of-words*

3.1.2. *Continuous skip-gram*

3.2. GloVe

3.3. *Embedding layer*

Referencias

[1] D. Karani, "Introduction to word embedding and word2vec." https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa, 2018.

[2] W. Koehrsen, "Neural network embeddings explained." https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526, 2018.

[3] J. Brownlee, "What are word embeddings for text?." https://machinelearningmastery.com/what-are-word-embeddings/, 2017.