Tarea 3

‘Búsqueda de Relaciones Sintácticas’

Word Embeddings
Genetic Algorithm

Alumno: Guillermo Pilleux
Profesor: Alexandre Bergel

Date: 13/11/17

Introducción

Para la realización de esta tarea, se continuará usando intensivamente el Word Embedding (WE) proporcionado por el profesor Jorge Pérez. Como se explicó en la tarea pasada, un WE es capaz de generar sentido de las palabras dentro de una oración. Vimos que a partir de un texto, se puede calcular el vector representativo de las oraciones que lo conforman para así obtener un vector que denota al texto propiamente tal.

De esta manera, el objetivo principal de esta tarea es aplicar un algoritmo genético a la búsqueda de la representación vectorial de una palabra dentro de un Word Embedding (WE). La palabra en cuestión, generará una relación entre tres otras palabras que se escogerán para que finalmente, al obtener la descendencia más “adaptada” de la población establecida, se comparará con una cantidad de palabras del WE para ver cuál fue la más parecida a la que se buscaba.

Procedimiento

Ya teniendo el WE almacenado en una base de datos MySQL con un indexamiento Sphinx para su rápida consulta, se explicara a continuación cómo se obtienen las relaciones entre las palabras y cómo encontramos las palabras más parecidas a la buscada.

Generación de vectores representativos

Para buscar en la base de datos cada vector representativo se utiliza PHP para lograr el uso de Sphinx, por lo que se escribe en un archivo de texto las palabras que se desean buscar, luego se le pasa al archivo PHP que busca cada palabra en la base y arroja un nuevo archivo de texto con los vectores representativos de cada palabra.

Relación

Se tomará el siguiente ejemplo.

"Rey es a Hombre como Reina es a Mujer" = Rey:Hombe :: Reina:Mujer

Lo que dice esta expresión es que la distancia entre Rey y Hombre tiene que ser la misma distancia vectorialmente (en sentido y magnitud) entre las palabras Reina y Mujer.

Como vimos que el vector representativo de una oración se puede calcular como el promedio de los vectores representantes de cada palabra que la compone, asumimos una postura de que la suma y resta de vectores representantes es válida para mantener estas relaciones.

Dicho esto, la fórmula resultante para el ejemplo anterior quedaría: Rey – Hombre – Reina + Mujer = 0

*También se puede tomar las distancias coseno entre los vectores, pero en la práctica de mi tarea, no arrojó resultados coherentes por lo que se omitirá.

Algoritmo Genético

La representación del individuo será un objeto de la clase Word, el cual tiene su fitnessValue y un vector para almacenar cada componente de su representación vectorial.

Esta misma clase tiene la capacidad de generar la población de Words solicitada por la clase WordGuesser.

WordGuesser es la clase que implementará todo el algoritmo genético.

Para cada individuo 'w' de la clase Word, sean w_1, w_2, w_3 tales que cumplen la relación

$w - \mathrm{w1} - \mathrm{w2} + \mathrm{w3} = 0$ (1) Vendría siendo las operaciones aritméticas sobre vectores.

La función de fitness de 'w' se calcula como sigue:

$$\forall w \in Word, f(w) = \frac{1}{w - \mathrm{w1} - \mathrm{w2} + \mathrm{w3}}$$

Cabe resaltar que el algoritmo buscará maximizar f(w) dado que se desea que el denominador sea cercano a 0.

Selección de individuo

Se implementa directamente la selección de la Roulette, esto es

1. Ordenar el vector de la población
2. Calcular el vector de fitness acumulado
3. Arrojar un valor al azar R
4. Se elige el último individuo cuyo fitness acumulado es más pequeño que R.

Reproducción

1. Escoger dos individuos según el método anterior que jugarán el rol de los padres
2. Al azar, escoger un 'cross over spot'
3. Copiar las celdas desde [0, crossOverSpot - 1] del primer individuo al hijo
4. Copiar las celdas desde [crossOverSpot, final] del segundo individuo al hijo
5. Arrojar un valor al azar que determina si habrán mutaciones
    a. Si las hay, intercambiar celdas al azar del hijo

Escoger individuo más apto

1. Tras un número determinado de iteraciones, ordenar la población según su fitness
2. Escoger el individuo con mayor fitness

Aplicación

Primero se busca una relación entre palabras y se escoge una de las palabras como la incógnita, despejando e igualando a 0. Para las tres palabras restantes, se buscan sus vectores representativos y se almacenan en un archivo de texto.

Segundo se ejecuta el MainGuesser que se le pasa como argumento el archivo con los vectores representantes de la expresión. Producirá un nuevo archivo de texto con el offspring resultante.

Tercero se genera un diccionario a partir del cual deseamos comparar, es decir, se busca una colección de palabras, sinónimos, antónimos y hasta palabras sin ninguna relación a la palabra en búsqueda para ver su comportamiento, almacenándose en otro archivo de texto. Este archivo se le entrega al código PHP que genera el vector representativo de cada palabra y también lo almacenará en un archivo de texto aparte.

Finalmente se ejecuta el main de la clase Evaluator que recibe como parámetros la cantidad de palabras del diccionario, el vector del offspring, el diccionario y los vectores del diccionario. Lo que devuelve esta clase es un archivo de texto con los resultados de las comparaciones.

Para cada ejemplo que se verá en la sección de resultados se tiene:

- Dictionary.txt: dentro de estas palabras se buscará la que más se parece a la que deseamos
- Expression.txt: X:1ra línea :: 2da línea:3ralínea (se omite la palabra 'X' que se busca)
- num_words.txt: cantidad de palabras en dictionary.txt
- offspring.txt: vector representante obtenido por el algoritmo genético
- results.txt: breve resumen de las palabras más parecidas y menos parecidas de dictionary
- similar_words.txt: vectores representativos de las palabras en expression
- vectors.txt: vectores representativos de dictionary

Resultados

1) X – hombre – Reina + Mujer = 0 (Se busca Rey)

Entre las palabras más parecidas se encuentran: Rey, Príncipe, beber, Reina y Roma.

Entre las palabras menos parecidas se encuentran: Dominante, pimienta, opresor.

Este ejemplo vendría siendo el más clásico y el que mejor resultados se obtuvieron. Vemos que se encuentra directamente la palabra Rey y que también encuentra sinónimos relativamente acertados a lo que se buscaba.

Dentro de las palabras menos parecidas, se encuentran dominante y opresor, lo que puede traer alguna noción de cómo está estructurado el WE, dependiendo de la perspectiva de quien lo evalúa (distintas personas pueden decir que los reyes fueron buenas personas con el pueblo o simplemente unos dominantes, opresores).

2) X – Santiago – China + Pekin = 0 (Se busca Chile)

Entre las palabras más parecidas se encuentran: Corrupción, Santiago, Chile, Bolivia.

Entre las palabras menos parecidas se encuentran: Guatemala, Uruguay, Politica.

Nuevamente, se logra encontrar la palabra buscada aunque no es la palabra que más se parece, sino que es corrupción (¿sospechoso?). Dentro de las palabras que menos se parecen no se logra encontrar alguna relación directa como en el ejemplo anterior.

3) X – {letra de un poema} – canción + {letra de una canción} = 0 (Se busca poema)

Entre las palabras más parecidas se encuentran: Justo, Tierra, religión, lírica.

Entre las palabras menos parecidas se encuentran: propósito, felicidad, sonrisas, poesía.

Este fue el ejemplo con peor resultado. Lo que se buscó hacer es en base a la letra de un poema, la palabra canción y la letra de una canción, deducir la palabra poema. Como vemos, dentro de las palabras parecidas se encuentra lírica, lo que puede ser algo acertado. También nos damos cuenta en la palabra Tierra, dado que el poema habla de rosas, cielo y estrellas puede existir alguna relación con esta palabra.

Observando las palabras menos parecidas, se logra ver que el resultado de esta relación fue un fracaso. Una de las palabras menos parecidas es exactamente la palabra que se buscaba, además de sinónimos del tópico principal del poema.