Tópicos selectos de Análisis de Datos
Tarea 3

Para entregar el 15 de octubre de 2018

1. Este ejercicio es sobre clasificación y análisis de sentimientos.

   Considera los datos que se encuentran en `spanish_reviews.zip`, que corresponden a opiniones de usuarios en los siguientes productos: automóviles, hoteles, lavadoras, libros, teléfonos celulares, música, computadoras y películas[1]. Para tu comodidad, he preparado dos conjuntos de datos: `train` (80 %) y `test` (20 %). Para cada uno de ellos hay dos categorias: `yes` y `no`, que indican las opiniones positivas y negativas, respectívamente.

   a) Implementa el clasificador ingénuo Bayesiano para las opiniones positivas y negativas. Usa los datos `train` y `test` para ajustar y verificar los resultados de tu clasificador, respectívamente.

   b) Ajusta clasificadores basados en SVM y otro de tu preferencia (CART, Boosting, NNet, etc...) usando la matriz de términos *pesada* con el criterio Term Frequency-Inverse Document Frequency (`TF-IDF`). Compáralo con el ingénuo Bayesiano.

   c) Obten representaciones vectoriales de las palabras en los documentos de las opiniones usando word embeddings con `word2vec`. Verifica si existen patrones evidentes entre palabras que corresponden a adjetivos, sustantivos y/o diferentes entidades encontradas. Describe cómo podrías usar esto para mejorar el clasificador ingénuo Bayesiano.

2. Este ejercicio es sobre semántica vectorial y word embeddings.

   En el archivo `spanish_billion_words.zip` se encuentra un Corpus en español que reúne corpus de diferentes fuentes [2].

   a) Utiliza `word2vec` para obtener representaciones semánticas vectoriales de las palabras del corpus. Verifica cualitativamente su desempeño escogiendo algunas *palabras clave*. Estas pueden ser arbitrarias o corresponder a ciertas etiquetas gramaticales. Indica el criterio que usaste para seleccionarlas. Usa gráficos informativos para ilustrar tus respuestas.

[1] tomados de `https://www.sfu.ca/~mtaboada/SFU_Review_Corpus.html`.

[2] Ver `http://crscardellino.github.io/SBWCE/`, pero tiene algunas modificaciones por mi parte.

b) Realiza clustering mediante $K-$means. Elige alguno(s) valor(es) K, Ilustra y comenta tus hallagos.

3. Este ejercicio es sobre Machine Translation (MT).

   El objetivo es implementar en forma simplificada, el paper de Mikolov et al.[3], el cual, a grandes rasgos, consiste en extender de forma automática diccionarios que puedan traducir palabras y frases de diferentes lenguajes usando representaciones vectoriales de palabras *dentro* de dichos lenguajes.

   El método es sencillo y sus pasos se describen a continuación.

   Considera un problema de traducción del un lenguaje fuente a otro lenguaje objetivo. En este ejemplo será del español al inglés.

   - Obtén embeddings monolinguales usando `word2vec` sobre un corpus adecuado para cada idioma, obteniendo así espacios vectoriales para el lenguaje fuente $\mathbb{A}$ y el objetivo $\mathbb{B}$.
   - Usa un diccionario bilingüe reducido para obteer un *mapeo* lineal entre los espacios vectoriales de ambos idiomas a través de una *Matriz de Traducción* $\mathbf{W}$, que obtienes resolviendo

     $$\min_{\mathbf{W}} \sum_{i=1}^{n} \|\mathbf{W}\mathbf{x}_i - \mathbf{z}_i\|^2,$$

     donde $\{\mathbf{x}_i, \mathbf{z}_i\}_{i=1}^{n}$ son los embeddings de los pares de palabras del diccionario que traducen $\mathbf{x}_i \rightarrow \mathbf{z}_i$. Estos los seleccionas de los embeddings que corresponden a los corpus monolinguales.
   - Para una nueva palabra $\mathbf{x} \in \mathbb{A}$, obten su mapeo mediante $\mathbf{z} \approx \mathbf{W}\mathbf{x}$, y obtén su *traducción* correspondiente buscando su embedding más cercano (distancia coseno) en el espacio del lenguaje objetivo $\mathbb{B}$.

   a) Implementa el procedimiento anterior usando los corpus que se encuentran en el archivo `europarl_es-en.zip`, que contiene corpus paralelos (español–inglés) extraídos de reuniones y debates del parlamento europeo[4]. Los corpus paralelos son versiones alineadas en dos idiomas de un mismo documento.

   Verifica el desempeño del método con algunos ejemplos ilustrativos de palabras como en el paper de Mikolov. ¿Son equivalentes los resultados? ¿Qué sugieres para mejorarlos?

[3] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. arXiv:1309.4168

[4] `http://www.statmt.org/europarl/`

Notas sobre los ejercicios.

1. En el ejercicio 1, puedes usar la librería que prefieras para construir las matrices de términos que necesites. Noté que hay detalles en algunos documentos (recuerda que son *datos generados por usuarios*). Uno de ellos es que algunas sentencias que terminan con "." o "," no tienen espacio antes de iniciar la siguiente (`[...]con el.Muy normal[...]`). Esto puedes resolverlo con expresiones regulares y reemplazos, por ejemplo, usando `gsub`. Si usas la librería `tm` de R, puedes usar funciones de transformación. Por ejemplo, la siguiente declaración resuelve el problema anterior:

   ```
   tm_map(corp,content_transformer(gsub),
                       pattern="([(a-zA-Z0-9)])\\.([a-zA-Z0-9])",
                       replacement="\\1. \\2")
   tm_map(corp,content_transformer(gsub),
                       pattern="([(a-zA-Z0-9)])\\.([a-zA-Z0-9])",
                       replacement="\\1, \\2")
   ```

   y también puedes usar ésta:

   ```
   tm_map(corp,content_transformer(gsub),
                       pattern="\\S+@\\S+",replacement="")
   ```

   que quita la información sobre emails que no es tan útil.

   Trata de obtener matrices de términos y documentos (TDM) lo menos "ralas" posibles, por ejemplo, con un parámetro de `sparsity` alrededor del 95 %.

   En la implementación de los demás métodos de clasificación (no el ingénuo Bayesiano), hay que incluir los documentos de prueba en la construcción de la TDM, a menos que guardes los tokens a contabilizar y los valores para pesarlos mediante `TF-IDF`.

   Guíate con el etiquetado universal de POS-tags para identificar las etiquetas de tu interés, una vez anotado el corpus. `http://universaldependencies.org/u/pos/`.

2. En el ejercicio 2, puedes usar las diferentes implementaciones de `word2vec`. Está el código original de Mikolov en `C++`, que está disponible en la página del curso. En `Python` puedes usar la librería `gensim`. En `R` puedes usar `rword2vec`, pero **recomiendo** usar `wordVectors` (`https://github.com/bmschmidt/wordVectors`), que instalas desde github. Las instrucciones y ejemplos muy ilustrativos del uso y funcionalidad de la librería está en la "vignette":

`https://github.com/bmschmidt/wordVectors/blob/master/vignettes/introduction.Rmd`.

Una de las utilidades te permite preparar tu corpus realizando el preproceso necesario mediante la función `prep_word2vec()`, así como encontrar palabras "cercanas" en el espacio del embedding usando distancia coseno `closest_to()`, además de visualizaciones apropiadas con PCA y otras funcionalidades.

Si tienes dudas, puedes consultarme (espero saber la respuesta...).

Ten mucho cuidado con el corpus que usaremos (spanish billion words), ya que es realmente grande (aunque quizá no tanto en el marco de Big Data. Lines: 46925295, Words: 1420665796, Bytes: 8810101297), y el proceso para `word2vec` puede generar un archivo muy grande, además que el tiempo de entrenamiento puede ser muy grande también. Puedes usar un sub-corpus pequeño, no necesariamente el completo, que contiene 100 subdocumentos.

3. En el ejercicio 3, lo recomendable es seguir el procedimiento del paper original para construir el diccionario bilingue, es decir:

   - Seleccionar las 5K palabras más frecuentes en el idioma fuente
   - Realizar su traducción usando Google Translate

   Sin embargo, dado el tamaño del corpus, construir una matriz de términos (por ejemplo, con `tm`) para verificar la frecuencia de palabras es bastante restrictivo. Puedes buscar versiones paralelizables que aceleren el tiempo de cómputo, o sino (como yo hice), construir un diccionario "alternativo", que obtuve a partir de uno inglés-italiano `OPUS_en_it_europarl_train_5K.txt`, que posteriormente traduje del inglés al español usando la librería `googleLanguageR`, que es una interfaz a la API de Google. En `https://cran.r-project.org/web/packages/googleLanguageR/vignettes/setup.html` hay instrucciones de instalación y uso. También puedes realizar la traducción "manualmente", pero te recomiendo usar la API de Google.

   Puedes usar otros corpus monolinguales o paralelos, por ejemplo, los existentes en el sitio `http://www.statmt.org/wmt11/translation-task.html`. Algo interesante es probarlo con distintos idiomas también.