TP de "Procesamiento del lenguaje natural mediante redes neuronales" - ECI 2019

Lucas Daniel Tomasini

Informe:

Utilicé como modelo el clasificador de fasttext, realizando previamente un preprocesamiento de los datos:

- Convertir todos los caracteres a minúscula
- Eliminar los puntos
- Despegar comas y puntos y comas de las palabras (no las eliminé ya que podrían aportar información a la hora de clasificar)
- Limpiar las contracciones del tipo “n’t” (ejemplo: “isn’t” = “is not”)
- Reemplazar guiones por espacios

También utilicé bi-gramas en lugar de uni-gramas para mejorar la performance del modelo (evaluando sobre los datos de desarrollo). Esta mejora se puede explicar en parte a las clasificaciones de las implicancias neutrales, ya que, como dice el paper, oraciones que contienen cláusulas de propósito (ej: “to catch a stick”) tienden a ser neutrales, y dichas cláusulas se modelan mejor usando bi-gramas (“to” + verbo).

Los hiperparámetros óptimos del modelo, salvo *wordNgram = 2*, resultaron ser los que vienen por default en el método *tran_supervised*.

El tamaño del vocabulario resultante luego de entrenar el modelo es 9842, y la precisión evaluada sobre los datos de desarrollo es del 67,82%.

Finalmente, uní los datos de entrenamiento y desarrollo en un único archivo y reentrené el modelo con ese nuevo conjunto, el cual utilicé luego para predecir los labels del conjunto de testing.

Ejecución del programa:

- Copiar todos los archivos fuente descargados de kaggle a la carpeta *snli_1.0* ubicada en el directorio raíz del proyecto.
- Correr de principio a fin la Jupyter Notebook
- En la carpeta *output_files* se generarán los archivos necesarios para entrenar y evaluar el modelo.
- El archivo *result.csv* se generará automáticamente en la carpeta raíz del proyecto.