TAREA CALIFICADA #2 INAR 2024

Modelos RNNs

El primer paso ha sido buscar información sobre qué modelos y/o métodos existían para la generación de texto, ya fuera para traducción, corrección ortográfica o simplemente para lo que se pedía en la tarea (generar texto a partir de un input).

Después de estar investigando me hice un listado con las que más se repetían por distintos foros y documentos: Vanilla RNN (modelo básico), LSTM, GRU, BRNN, Stacked RNN/LSTM/GRU, Seq2Seq y Transformers.

Me puse a probar con el modelo de Abascal y Casado los diferentes modelos y variando capas para ver que resultados me arrojaban cada uno.

La primera conclusión que saqué es que Transformers no me iba a servir, ya que al descargar el modelo, este importaba rasgos y el modelo terminaba hablando en inglés, por lo que no se realizaba un entrenamiento puro con el dataset proporcionado.
BRNN en teoría iba a ser de los mejores modelos para entrenar según diversas opiniones y documentación al respecto, pero en mis pruebas me di cuenta que este modelo terminaba destrozando todo y no conseguí hacer que generará algo con un mínimo nivel.
Por otro lado, el modelo GRU si me dió modelos que estaban al nivel (un poco por debajo) de los modelos LSTM que generé previamente, pero el tiempo de entrenamiento llegaba a ser incluso 4 veces superior para llegar a un nivel equivalente.

Por todo lo anterior, decidí quedarme con el modelo LSTM para entrenar a los 3 parlamentarios.
Aún así tuve que generar diversos modelos LSTM para llegar a la mejor optimización que veía capas con el dataset de cada uno.
Algo importante que me di cuenta con el modelo de Abascal fue que el bajar el BathSize no le iba bien, pues perdía toda la referencia y generaba texto con palabras aleatorias. Tuve que realizar los modelos con múltiples capas y con un BathSize lo suficientemente grande para que generara algo con contexto.
En los modelos con un dataset pequeño el modelo se comportaba mejor con un número de capas de entrenamiento superior que los modelos entrenados con un dataset más grande.

En conclusión, los modelos que me quedé como finales son: Abascal10, Casado12 y Sanchez09.

Los notebook finales ya los fui preparando para que me guardara el modelo en .h5 o .keras.

Benchmark

Para evaluar la calidad del texto generado, decidí utilizar varias métricas y enfoques.
- Perplejidad: La perplejidad mide la probabilidad de que un modelo haya generado un conjunto de palabras dado un contexto. Un modelo con baja perplejidad produce textos más coherentes.
- BLEU: Mide la similitud entre las n-gramas generadas por el modelo y un conjunto de referencias.
- ROUGE: Evalúa la coincidencia de las n-gramas entre el texto generado y el texto de referencia.

En un principio generé un código que evaluaba el texto generado sin tener en cuenta el contexto. Decidí cambiar esto para que el benchmark se realizará de forma que se le diera a cada modelo el inicio de una frase que estuviera en el dataset de entrenamiento y comparar que generaba el modelo con lo que realmente continuaba diciendo cada parlamentario.
Por último, para que fuese más preciso a la hora de calificar los resultados, cada modelo generaría tres textos a partir de 3 inicios diferentes, para así después calcular la media.

En total hay 3 notebook: benchmarkAbascal, benchmarkCasado y benchmarkSanchez.

Cada notebook pide el modelo entrenado (.h5 o .keras) y el txt para generar el tokenizador.

Diálogo

La última parte fue ponerlos a dialogar. Una vez teniendo el modelo que preparé a cada uno para realizar el Benchmark, me basé en esa estructura con la que se generaba el texto para así ponerlos a dialogar.

El diálogo se basa en 5 rondas (Sánchez, Casado y Abascal; en ese órden), donde en la primera interacción de Sanchez se le da una frase aleatoria de su propio dataset.
Cada interacción generará 75 palabras teniendo en cuenta las últimas 30 palabras de la anterior intervención.

El diálogo se encuentra en el notebook: dialogoCongreso.