Jose Arias Moncho
Victoria Beltrán Domínguez

LINGÜÍSTICA COMPUTACIONAL MEMORIA PRÁCTICA MODELOS DE LENGUAJE

Septiembre 2021

Índice

Introducción

En esta práctica, se plantea construir y evaluar distintos modelos de lenguaje utilizando el toolkit *SRILM*. Particularmente, partiendo del corpus *Dihana*, se realizará una serie de experimentos para detectar cómo afecta el valor N de los N-gramas, el método de suavizado y el método de descuento a la perplejidad del modelo. Además, utilizando el corpus *Europarl*, también se estudiará cómo varía la selección del vocabulario de entrenamiento del modelo a la perplejidad del modelo. Pasamos a detallar cómo se han solventado cada una de las tareas, así como qué resultados hemos obtenido.

1. Tarea 1: Comparación de los diferentes modelos de lenguaje según el valor de N.

En esta primera tarea, manipulando el corpus *Dihana* (*dihana.entrenamiento1.txt* y *dihana.prueba1.txt*), se pide construir y evaluar diferentes modelos de n-gramas variando el valor de N. Por el momento, utilizaremos el resto de parámetros por defecto.

Con el fin de comparar cómo afecta la variación de N en la perplejidad vamos a construir los diferentes modelos, entrenándolos a partir del conjunto de entrenamiento dado. Para realizar esto en SRILM vamos a utilizar la siguiente orden en la línea de comandos, donde N corresponde con el número del n-grama deseado:

```
ngram-count -order N -lm modelo_N -text
dihana.entrenamiento1.txt
```

Una vez obtenido el modelo entrenado, vamos a evaluarlo con el conjunto de datos de test, obteniendo la perplejidad del modelo. Esto se ha conseguido con la siguiente orden:

```
ngram -order N -lm modelo_N -ppl dihana.prueba1.txt >
perplejidad_N.txt
```

Realizando este proceso para diferentes modelos de n-gramas, dado que $N = \{1, 2, 3, 4, 5\}$, obtenemos las perplejidades reflejadas en la siguiente tabla:

N-grama	Perplejidad
1	**97.05**
2	**11.22**
3	**7.68**
4	**7.23**
5	**7.27**

Los resultados obtenidos para esta tarea muestran que a medida que se incrementa el valor N del N-grama, la perplejidad va disminuyendo hasta que para $N = 5$, la tendencia deja de disminuir y vuelve a aumentar. De esta manera, consideraremos que los modelos basados en trigramas y 4-gramas son los que mejores prestaciones nos dan, puesto que el modelo de 5-gramas, a pesar de que aumentamos la complejidad del modelo no consigue disminuir la perplejidad de este.

2. Tarea 2: Comparación de modelos de lenguaje según el método de descuento utilizado.

En esta segunda tarea, vamos a utilizar nuevamente el corpus *Dihana* (*dihana.entrenamiento1.txt* y *dihana.prueba1.txt*) para construir y evaluar modelos de trigramas y 4-gramas variando el valor de método de descuento utilizado. Específicamente, probaremos con los métodos de descuento *Good-Turing*, *Witten-Bell*, *modified Kneser-Ney* y *unmodified Kneser-Ney*.

Indagando en la documentación de ngram-count obtenemos la información necesaria para poder realizar el entrenamiento variando el método de descuento. Concretamente, dado un valor N, utilizaremos el argumento `-wbdiscount`*N* para *Witten-Bell*, `-kndiscount`*N* para *modified Kneser-Ney* y `-ukndiscount`*N* para *unmodified Kneser-Ney*. Por defecto, el comando utiliza el descuento Good-Turing, por lo que para utilizarlo no tenemos que añadir nada.

Realizando los experimentos pertinentes utilizando $N = \{3, 4\}$ obtenemos las perplejidades reflejadas en la siguiente tabla:

Método de descuento	**N-grama**	**Perplejidad**
Good-Turing	**3**	**7.67**
	4	**7.22**
Witten-Bell	**3**	**7.91**
	4	**7.20**
Modified Kneser-Ney	**3**	**8.14**
	4	**7.61**
Unmodified Kneser-Ney	**3**	**7.68**
	4	**7.10**

Como podemos observar en la tabla anterior, para este conjunto de datos de entrenamiento y test, el modelo que obtiene mejores resultados es el 4-grama con descuento *unmodified Kneser-Ney*. Por otro lado, comparando los trigramas podemos comprobar que utilizando un descuento *Good-Turing* se obtiene la mejor perplejidad. Por último, en este caso podemos identificar claramente que los modelos que utilizan el descuento de *modified Kneser-Ney* obtienen peores resultados para todos los valores de N probados.

3. Tarea 3: Comparación del suavizado por backoff e interpolación.

Esta tercera tarea es la última en la que trabajaremos con el Corpus *Dihana* (dihana.entrenamiento1.txt y dihana.prueba1.txt). En esta tarea se pide construir y evaluar modelos de trigramas y 4-gramas, utilizando interpolación como método de suavizado y variando el método de descuento utilizado entre *Witten-Bell* y *modified Kneser-Ney*.

Ya hemos indagado sobre cómo utilizar los diferentes descuentos indicados, pero no sabemos cómo utilizar interpolación para suavizar el modelo. Para ello, tendremos que añadir a la orden de entrenamiento el argumento `-interpolate`*N*, siendo N el tamaño del modelo de N-gramas.

Realizando los experimentos obtenemos el siguiente desempeño de las diferentes configuraciones de los modelos para estos datos:

Método de suavizado	**Método de descuento**	**N-grama**	**Perplejidad**
Interpolación	**Witten-Bell**	**3**	**7.54**
		4	**6.82**
	Modified Kneser-Ney	**3**	**7.50**
		4	**6.83**
Backoff	**Witten-Bell**	**3**	**7.91**
		4	**7.20**
	Modified Kneser-Ney	**3**	**8.14**
		4	**7.61**

Utilizando los resultados anteriores, podemos indicar que con este corpus de entrenamiento y test, el método de suavizado de interpolación obtiene mejores resultados en comparación al suavizado backoff. Además, el mejor modelo se obtiene con un 4-grama con suavizado de interpolación y

descuento *Witten-Bell*, siendo el siguiente mejor la misma configuración, pero con un descuento de modified *Kneser-Ney*, que únicamente tiene una diferencia de perplejidad del 0.01 con respecto al mejor.

4. Tarea 4: Modelos de lenguaje para el corpus Europarl con diferentes tallas de vocabulario.

En esta cuarta y última tarea, vamos a trabajar con el corpus *Europarl*, construyendo y evaluando modelos de trigramas y 4-gramas y utilizando diferentes conjuntos de vocabulario en la etapa de entrenamiento del modelo. Específicamente, vamos a trabajar con un subconjunto del vocabulario original en el que eliminaremos ciertos términos atendiendo a su frecuencia en los datos de entrenamiento.

Tras realizar un proceso de preparación del corpus de entrenamiento y los conjuntos de vocabulario con los que se trabajará siguiendo las indicaciones de la práctica, utilizaremos el argumento `-vocab vocab.txt` para indicar qué vocabulario queremos utilizar, siendo `vocab.txt` el fichero con el vocabulario deseado a la hora de entrenar al modelo.

Los resultados obtenidos se presentan en la siguiente tabla:

Vocabulario	**N-grama**	**Perplejidad**
Palabras con frecuencia > 1	**3**	**83.18**
	4	**75.52**
Palabras con frecuencia > 5	**3**	**80.58**
	4	**73.15**
Palabras con frecuencia > 9	**3**	**79.00**
	4	**71.70**

De esta tabla podemos deducir que este conjunto de datos es más complejo, de tal forma que los todos los modelos obtienen perplejidades de un orden más que con el corpus *Dihana*. Si nos centramos en este conjunto de datos concretos, se observa un descenso de la perplejidad conforme se disminuye el vocabulario de entrenamiento. Esto podría ser a causa de que tenemos un corpus tan grande y estas palabras son tan poco comunes que eliminándolas del vocabulario ayudamos al modelo a obtener mejores representaciones de los N-gramas presentes y más frecuentes.

5. Conclusiones

En esta práctica hemos trabajado con la herramienta SRILM para entrenar y evaluar diferentes modelos de lenguaje partiendo de los dos corpus que se nos han proporcionado (*Dihana* y *Europarl*). A continuación, gracias a los conocimientos obtenidos a partir de la realización de las diferentes tareas, procedemos a finalizar esta memoria con algunos apuntes generales.

Como conclusiones relativas a los corpus utilizados, centrándonos en los modelos de N-gramas, hemos observado que los modelos de 4-gramas suelen obtener mejores resultados que el resto de N-gramas con un valor inferior y superior, siendo este el valor óptimo dentro de los modelos de N-gramas. Además, a pesar de no obtener resultados claramente superiores, el *unmodified Kneser-Ney* parece ser el mejor método de descuento. Por último, el suavizado de interpolación ha obtenido mejores resultados que el suavizado de backoff en la tercera tarea. Cabe destacar que estas conclusiones son muy relativas a los corpus utilizados, y que por lo tanto no consideramos que puedan extrapolarse a los diferentes corpus y modelos existentes.

Como conclusiones generalizables a otras tareas, podemos resaltar dos. La primera, relativa a los modelos de lenguaje, es que los modelos más complejos no tienen por qué dar los mejores resultados, algo que hemos comprobado tras realizar la primera tarea. La segunda y última conclusión es relativa al vocabulario utilizado en la fase de entrenamiento del modelo de lenguaje. Hemos comprobado en la última tarea la importancia del vocabulario de entrenamiento cuando trabajamos con una cantidad de datos más elevada, y entendemos que trabajar con vocabularios restringidos a las palabras más comunes puede mejorar el desempeño de los modelos entrenados.