Segmentación de Componentes Argumentativas con modelo CNN-BLSTM-CRF

Luis Ernesto Ibarra Vázquez[1], Luis Enrique Dalmau Coopat[2], and Adrián Hernández Pérez[3]

Universidad de La Habana

Resumen El Minado de Argumentos consiste en la extracción de componentes argumentativas y sus relaciones además de su posterior clasificación. El trabajo implementa y usa una red neuronal artificial con arquitectura CNN-BLSTM-CRF para la segmentación de las componentes argumentativas usando keras y tensorflow. El problema fue modelado como un problema sequence to sequence en el cual se quería conocer las etiquetas BIOES del texto para delimitar las componentes.

Keywords: Convolutional Neural Networks · Recurent Neural Networks · Argument Mining

1. Introducción

El Minado de Argumentos es la rama del Procesamiento del Lenguaje Natural encargada de la extracción de estructuras argumentativas de fuentes no estructuradas de texto. Las estructuras argumentativas en dicho problema se modelan como un grafo en el cual los nodos son las componentes argumentativas y las aristas las relaciones entre estas componentes. Los nodos del grafo se pueden clasificar en dependencia del tipo de componente que sea, como por ejemplo, afirmación o evidencia y además de la clasificación de los nodos se clasifican las relaciones entre estos, como por ejemplo en ataque, apoyo.

Existen enfoques sobre el Minado de Argumentos que asumen que las componentes argumentativas ya están extraídas y se enfocan sobre las demás partes del problema. En la práctica esto no se cumple, ya que el texto viene sin procesar haciendo estos enfoques trabajosos. En este trabajo se espera resolver este problema proveyendo de una implementación de un segmentador de componentes argumentativas. Se propone el problema como uno sequence to sequence con la arquitectura CNN-BLSTM-CRF. Se utilizará keras y tensorflow para dicha implementación.

2. Problema a resolver

El problema a resolver consiste en dado un texto devolver sus estructuras argumentativas. Para esto se modeló el problema como un problema sequence

to sequence, en el cual dado una secuencia de palabras se desea conocer sus etiquetas BIOES. Las etiquetas BIOES delimitan las componentes argumentativas haciendo posible su extracción. Se eligió este tipo en particular de etiquetas ya que se trabaja con BLSTM y con estas no se pierde información al leerlas de izquierda a derecha o de derecha a izquierda.

3. Modelos

3.1. Embeddings

Para la solución del problema se propone una arquitectura CNN-BLSTM-CRF. Para la representación de las palabras en la red neuronal artificial se utilizan embeddings GloVe con dimensión 300. La entrada de la red neuronal artificial entonces constituye en un tensor de dimensión $Lx300$ donde L representa el tamaño máximo de la secuencia a entrenar (en el caso de este trabajo se utilizaron oraciones). Secuencias que no tengan dicho tamaño se le hace padding a la derecha. Estos embeddings no son entrenables.

Dado que la arquitectura extrae información sobre los caracteres de las palabras, se necesita una forma de representar estos. La representación de cada caracter consiste en un embedding entrenable de dimensión 100, por lo que cada palabra estaría representada por un tensor de dimensión $Wx100$ donde W representa la longitud máxima de las palabras, a las cuales se les realiza padding en caso de no tener la longitud máxima.

3.2. Convolución

Estudios previos han demostrado que CNN es un enfoque eficaz para extraer información morfológica (como el prefijo o el sufijo de una palabra) de los caracteres de las palabras y codificarla en representaciones neuronales. La Figura 1 muestra la CNN usada para extraer la representación a nivel de caracter de una palabra dada. La CNN es similar a la de Chiu y Nichols (2015), excepto que se usan solo embeddings de caracteres como entradas a la CNN, sin características de tipo de carácter. Se aplica una capa de dropout antes de que se ingresen las embeddings de caracteres en CNN.

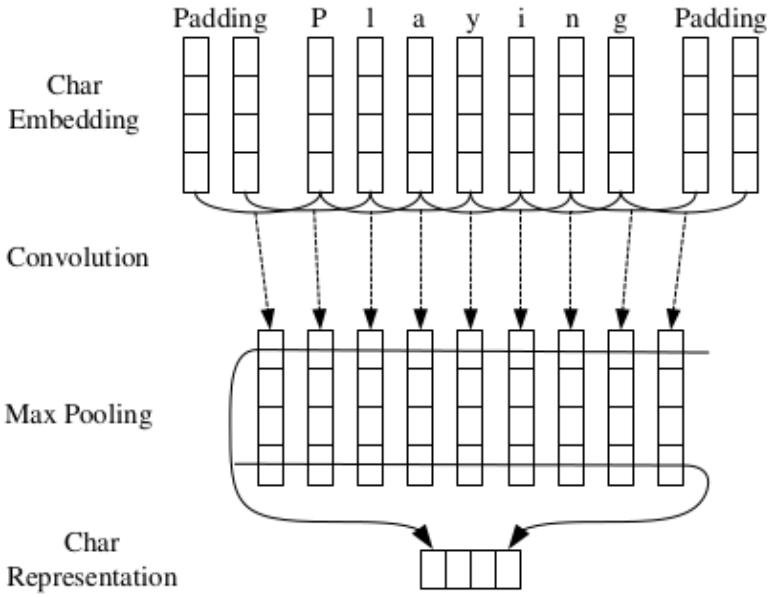

Figura 1. La red neuronal de convolución para extraer representaciones de palabras a nivel de carácter. Las flechas discontinuas indican una capa de dropout aplicada antes de que se ingresen los embeddings de caracteres en CNN.

Para esta capa se tomó el tamaño del kernel de convolción como 3, con padding hacia los lados no afectando así la dimensión del tamaño de la palabra al salir de la convolución. Como resultado se obtiene un vector de $W \times 30$ donde 30 es la cantidad de filtros de la capa. Esta salida a su vez luego es comprimida con un max pooling devolviendo la representación final de la palabra como un vector de dimensión 1×30 que será concatenada a la representación de la palabra asociada.

3.3. LSTM

Las redes neuronales recurrentes (RNN) son una poderosa familia de modelos conexionistas que capturan la dinámica del tiempo a través de ciclos en el grafo. Aunque, en teoría, las RNN son capaces de capturar dependencias de larga distancia, en la práctica fallan debido a los problemas de desaparición/explosión del gradiente . Los LSTM son variantes de los RNN diseñados para hacer frente a estos problemas de desaparición de gradientes. Básicamente, una unidad LSTM se compone de tres puertas multiplicativas que controlan las proporciones de información para olvidar y pasar al siguiente paso de tiempo. La Figura 2 muestra la estructura básica de una unidad LSTM.

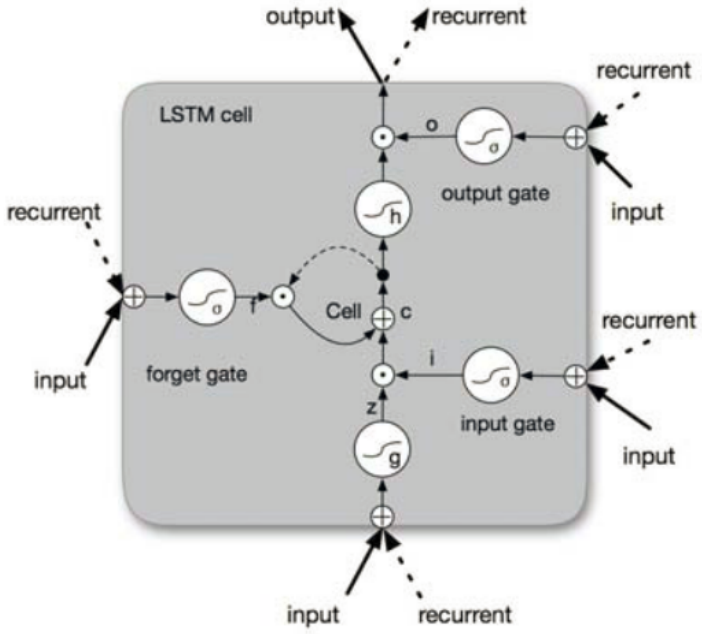

Figura 2. Esquema de una unidad LSTM.

Formalmente las fórmulas para actualizar una unidad LSTM en un tiempo t son:

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{1}$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \tag{2}$$

$$\tilde{c}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{4}$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

donde σ es la función sigmoidal elemento a elemento y $\odot$ es el producto elemento a elemento. x_t es el vector entrada (e.g. embeddings de palabras) en el tiempo t, y h_t es el vector de estado oculto (tambien llamado vector de salida) que guarda toda la información importante en (y antes) del tiempo t. U_i , U_f , U_c , U_o denota las matrices de pesos de puertas diferentes para la entrada x_t , y W_i , W_f , W_c , W_o son las matrices de pesos para el estado oculto h_t. b_i , b_f , b_c , b_o denotan los vectores de bias.

BLSTM Para muchas tareas de etiquetado de secuencias, es beneficioso tener acceso a contextos pasados (izquierda) y futuros (derecha). Sin embargo, el estado oculto de LSTM h_t toma información solo del pasado, sin saber nada sobre el futuro. Una solución elegante cuya eficacia ha sido probada por trabajos anteriores es LSTM bidireccional (BLSTM). La idea básica es presentar cada secuencia hacia adelante y hacia atrás en dos estados ocultos separados para capturar información pasada y futura, respectivamente. Luego, los dos estados ocultos se concatenan para formar la salida final.

3.4. CRF

Para las tareas de etiquetado de secuencias (o predicción estructurada general), es beneficioso considerar las correlaciones entre las etiquetas en los vecindarios y decodificar conjuntamente la mejor cadena de etiquetas para una oración de entrada dada. Por ejemplo, en el etiquetado POS, es más probable que un adjetivo vaya seguido de un sustantivo que de un verbo, y en NER con anotación BIO2 estándar, I-ORG no puede seguir a I-PER. Por lo tanto, modelamos la secuencia de etiquetas de forma conjunta utilizando un campo aleatorio condicional (CRF), en lugar de decodificar cada etiqueta de forma independiente.

Formalmente, usamos $z = \{z_1 , \cdots , z_n\}$ para representar una secuencia de entrada genérica donde z_i es el vector de entrada de la i-ésima palabra. $y = \{y_1 , \cdots , y_n\}$ representa una secuencia genérica de etiquetas para z. $Y(z)$ denota el conjunto de posibles secuencias de etiquetas para z. El modelo probabilístico para la secuencia CRF define una familia de probabilidad condicional $p(y|z; W, b)$ sobre todas las posibles secuencias de etiquetas z", dado "zçon la siguiente forma:

$$p(y|z; W, b) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, z)}{\sum_{y' \in Y(z)} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, z)} \tag{7}$$

donde $\psi_i(y', y, z) = exp(W^T_{y',y} z_i + b_{y',y})$ son funciones potencia, y $W^T_{y',y}$,$b_{y',y}$ son el vector de pesos y bias correspondiente al par anotado (y' , y), respectivamente. Para el entrenamiento de la CRF , se usa la estimación de la máxima verosimilitud condicional. Para un conjunto de entrenamiento (z_i, y_i), el logaritmo de la verosimilitud (conocido como la log-verosimilitud) esta dada por:

$$L(W, b) = \sum_i \log p(y|z; W, b) \tag{8}$$

El entrenamiento de la máxima verosimilitud escoge parámetros tal que la log-verosimilitud es maximizada.

Decodificar es buscar la secuencia de la etiqueta y^* con la mayor probabilidad condicional:

$$y^* = argmax_{y \in Y(z)} p(y|z; W, b) \tag{9}$$

Para un modelo CRF secuencial (solo se consideran las interacciones entre dos etiquetas sucesivas), el entrenamiento y la decodificación se pueden resolver de manera eficiente adoptando el algoritmo de Viterbi.

3.5. BLSTM-CNNs-CRF

Finalmente, se construye este modelo de red neuronal alimentando los vectores de salida de BLSTM en una capa CRF. La Figura 3 ilustra la arquitectura de esta red en detalle. Para cada palabra, la CNN calcula la representación a nivel de carácter en la Figura 1 con embeddings de caracteres como entradas. Luego, el vector de representación a nivel de carácter se concatena con el vector de de palabras para alimentar la red BLSTM. Finalmente, los vectores de salida de BLSTM se pasan a la capa CRF para decodificar conjuntamente la mejor secuencia de etiquetas. Como se muestra en la Figura 3, las capas de exclusión se aplican tanto en los vectores de entrada como de salida de BLSTM.

4. Implementación

Para la implementación de la red neuronal artificial se utilizó la imagen de docker *tensorflow/tensorflow:2.9.1-jupyter*. Como su nombre menciona, esta imagen trae la versión especificada de tensorflow y además está preparada

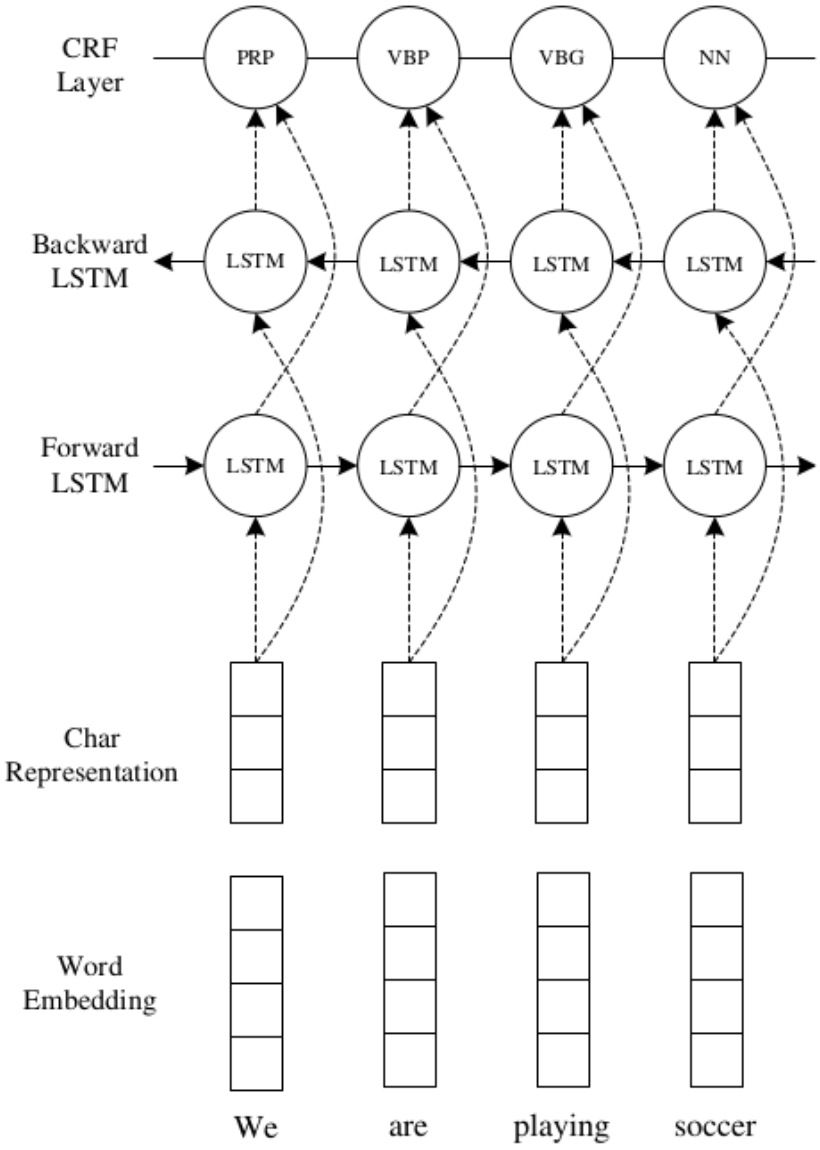

Figura 3. La principal arquitectura de la red neuronal. La representación de los caracteres de cada palabra son computados por la CNN en Figura 1. Luego la representación es concatenada con el embedding de la palabra antes de pasarselo a la red BLSTM. Líneas discontinuas indican que capas de dropout son aplicadas en la entrada y salida de BLSTM.

para ser corrida como un jupyter notebook. Esta imagen no contenía la implementación de CRF por lo que se creó otra a partir de esta con la biblioteca que lo contenía.

El componente principal del modelo constituyen las capas de este. Para la implementación se hizo uso de las siguientes [1] [2]:

- Vectorizer: Convierte las secuencias en vectores donde cada elemento representa el índice de la palabra en un vocabulario.
- Embedding: Capa usada para la conversión de vectores de índices en tensores conteniendo la representación final del elemento.
- Conv1D: Usada para realizar la convolución sobre las secuencias de embeddings de caracteres
- MaxPooling1D: Usada para calcular la representación final de la palabra basados en los resultados de la convolución
- Dropout: Usada para prevenir el overfitting en el conjunto de entrenamiento y permitir una correcta generalización del problema
- Bidirectional LSTM: Permite que las secuencias se apliquen a la capa LSTM en ambas direcciones.
- CRF: Calcula una distribución de probabilidad conjunta para clasificar secuencias con las etiquetas dadas.

5. Entrenamiento

5.1. Corpus Usado

El corpus usado en el entrenamiento constituye una versión en formato Conll del corpus Argument Annotated Essays Corpus [4]. Este corpus contiene una colección de 402 ensayos argumentativos. Además viene con una selección de archivos para entrenamiento, prueba y validación. Dicho corpus fue segmentado en oraciones, dando como resultado una división de 5096, 1452 y 609 oraciones para entrenamiento, prueba y validación respectivamente.

Para el entrenamiento se hizo preprocesamiento sobre el corpus inicial. Primero se llevaron las etiquetas de BIO a BIOES, luego se separaron las oraciones de los párrafos y se guardaron en un archivo con una oración en cada línea. Se hizo algo similar con las anotaciones de la oración. Además se crearon archivos conteniendo el vocabulario, los caracteres y las etiquetas para futuros procesamientos.

5.2. Hiperparámetros y optimización

La selección de los hiperparámetros del modelo fue basada en las sugerencias dadas en [3].

- Droput rate: 0.5
- Filtros: 30
- Tamaño de kernel: 3
- Cantidad de unidades en LSTM: 200

5.3. Duración

El modelo se corrió durante 25 epoch en un i5-6200U CPU @ 2.30GHz. Cada época duró aproximadamente 2 minutos, con un total de 50 minutos de entrenamiento.

6. Evaluación

Para la evaluación se tomaron las medidas de accuracy y de la función de pérdida y se crearon gráficas con los valores para ser examinados. Para tener más referencias se crearon modelos alternativos al propuesto. Entre estos se encuentran variantes del modelo sin la capa de embeddigns de caracteres y variantes que conectan a la salida de la capa BLSTM una red densa con la capa CRF. Las variantes tendrán las siguientes signaturas, CNN si tiene la capa de convolción de los caracteres y DL si tiene la capa densa.

6.1. Resultados

Cuadro 1. Resultados.

Arq.	Accuracy	Loss
CNN-BLSTM-DL-CRF	0.9097	0.6028
CNN-BLSTM-CRF	0.9080	0.5299
BLSTM-DL-CRF	0.8868	0.7424
BLSTM-CRF	0.8923	0.8415

Como se observa en la Tabla 1 la arquitectura con la capa de convolución presenta mejor desempeño en todos los aspectos. Las arquitecturas basadas en CNN contienen un accuracy semejante, aunque su diferencia en la pérdida es un poco más grande dado el uso o no de las capas densas.

Accuracy Como se puede evidenciar en la gráfica, a partir de 3 epochs el accuracy del modelo en validación se mantiene cercano al 90 % sin muchos cambios y en entrenamiento va hacia la sobreadecuación a partir de ese punto coincidente de ambas mediciones. Es recomendable unos 5 epochs que es lo más estable luego de la intersección.(Figura 4)

Loss Concordando con el análisis anterior la pérdida de Cross Entropy despega de manera acelerada luego de la intersección de ambas mediciones(entrenamiento y validación) en aproximadamente 3 epochs. De igual manera se recomienda utilizar 5 epochs para mayor certeza.(Figura 5)

Aquí también se pone en evidencia la presencia de sobreadecuación a los datos a medida que se usan muchos epochs para el entrenamiento

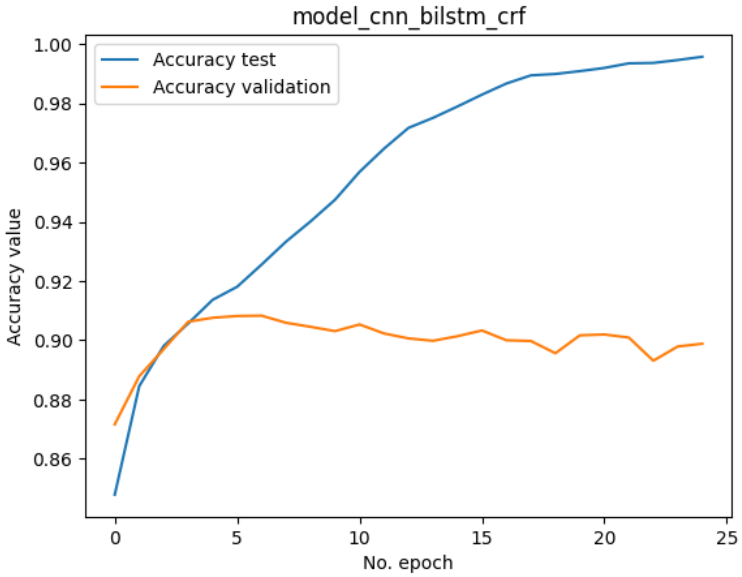

Figura 4. Accuracy durante entrenamiento CNN-BLSTM-CRF

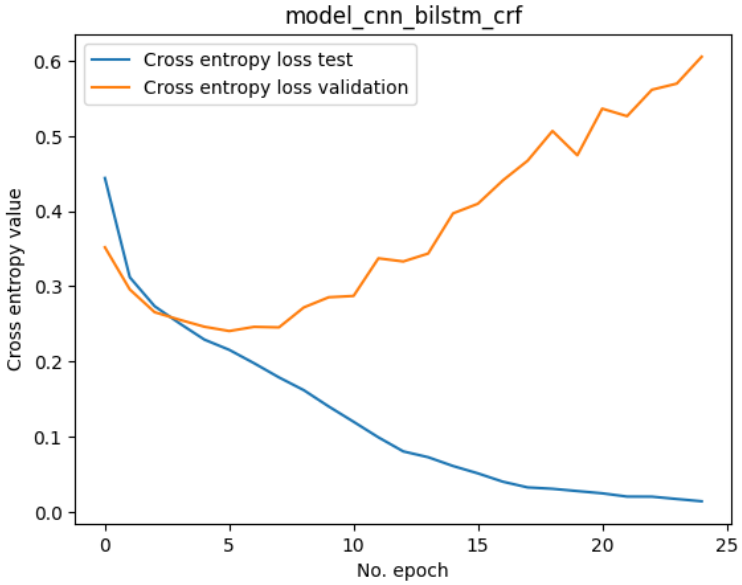

Figura 5. Pérdida durante entrenamiento CNN-BLSTM-CRF

7. Conclusiones

El modelo propuesto resuelve el problema con una accuracy aproximada del 90 %. Este resultado satisface las métricas deseadas del problema de segmentación de componentes argumentativas.

Referencias

1. TensorFlow Documentation `https://www.tensorflow.org/api_docs` 8
2. Keras Documentation, `https://faroit.com/keras-docs/1.2.0/` 8
3. Ma, Xuezhe and Hovy, Eduard: End-to-end sequence labeling via bi-directional lstm-cnns-crf: arXiv preprint arXiv:1603.01354 . 2016 8
4. Christian Stab and Iryna Gurevych. 2016. Parsing Argumentation Structure in Persuasive Essays. arXiv preprint 8