



# Neural Attention

what it is and  
what to do with it

Milagro  
Teruel

Tensorflow  
Meetup

Buenos  
Aires  
2018

# Milagro cuánto?

Estudiante de doctorado

+ Representation Learning (embeddings)

+ Educational Data Mining

Argument mining

+ INRIA Sophia Antipolis



# Sobre qué vamos a charlar hoy?



Disclaimer

Variedad de attention mechanisms

- + Cómo se puede aplicar atención dependiendo el tipo de red
- + Un poco más de detalle sobre sequence labeling

Implementación!

- + Armar un esqueleto de red con atención en Keras

PART I

# Neural Attention

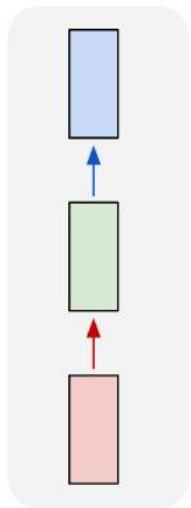


Weight something with  
attention scores

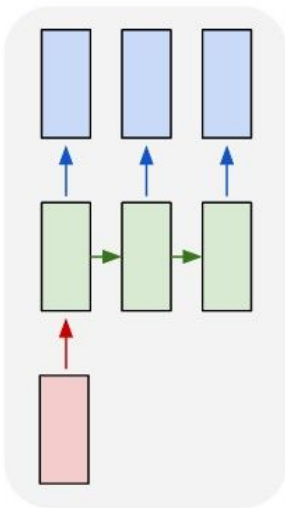


# Elegimos el tipo de red

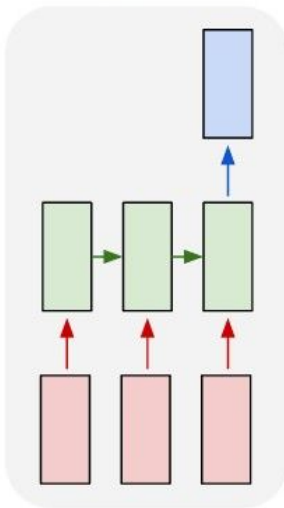
one to one



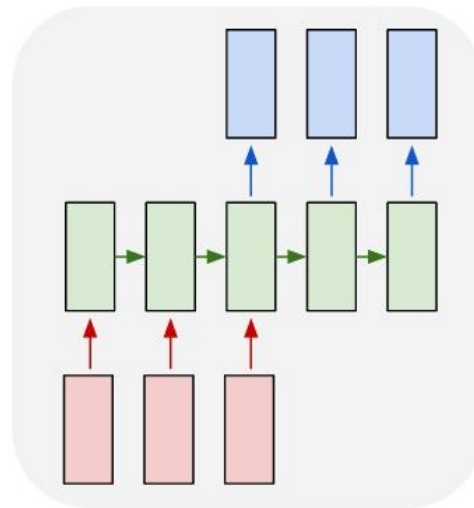
one to many



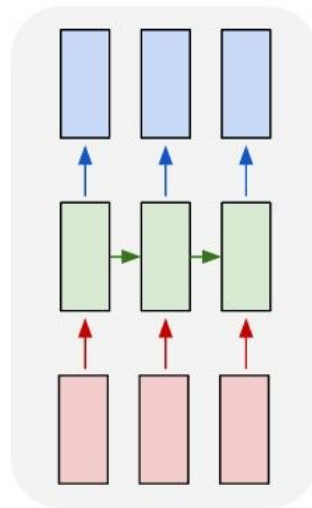
many to one



many to many

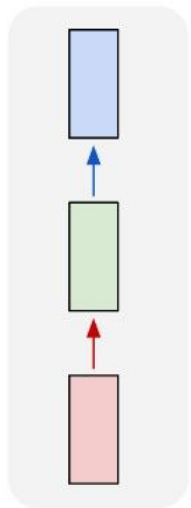


many to many

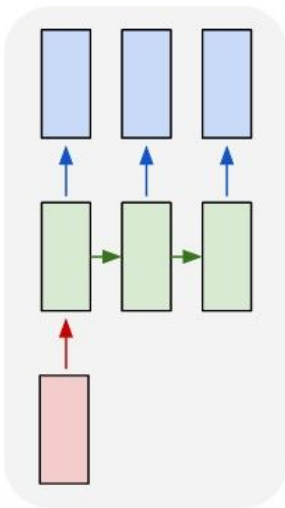


# Elegimos el tipo de red

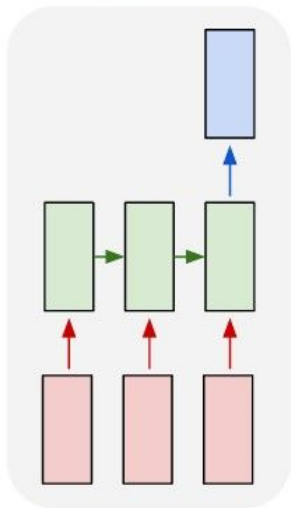
one to one



one to many

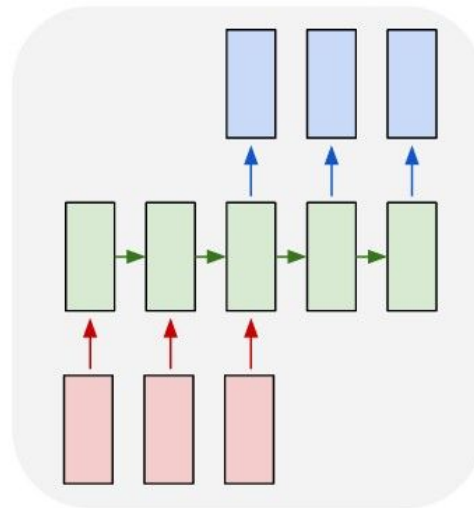


many to one



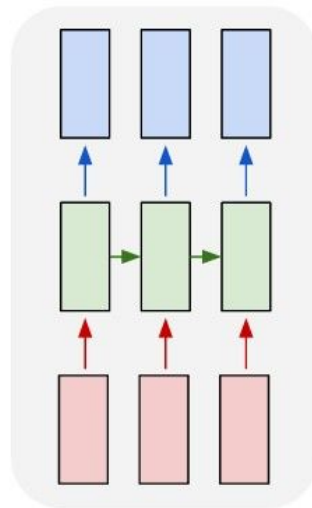
sequence  
classification

many to many



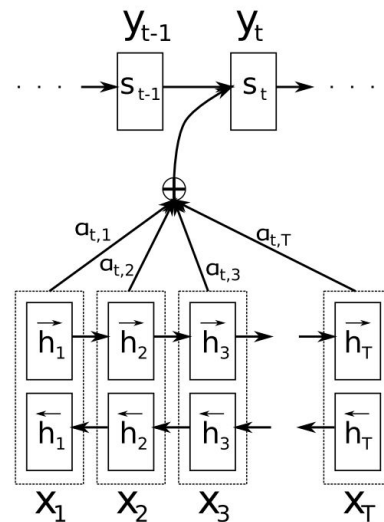
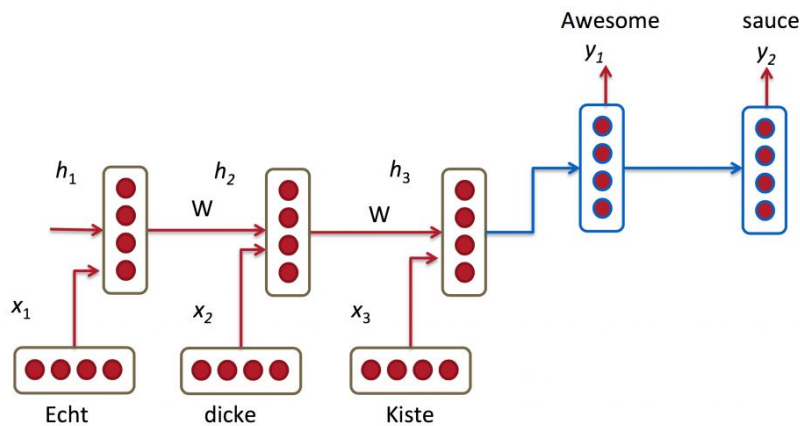
sequence  
to sequence

many to many



sequence  
labeling

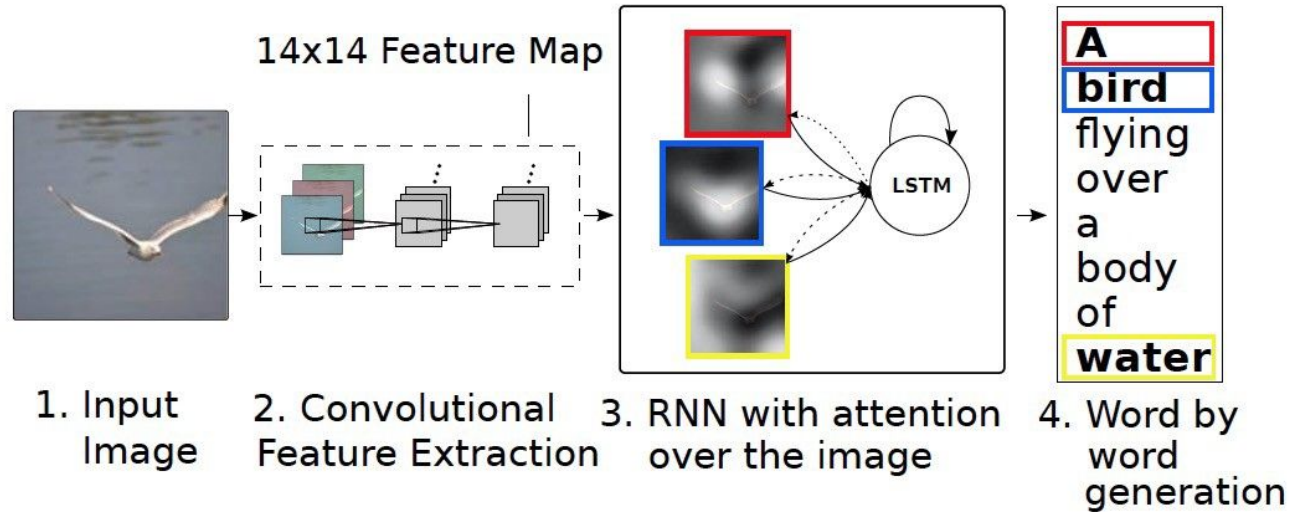
# Neural attention for machine translation [2]



Para generar cada palabra en el idioma target, pesamos todos los **recurrent output** de la red.



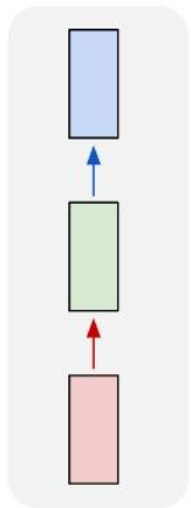
# NA for image captioning



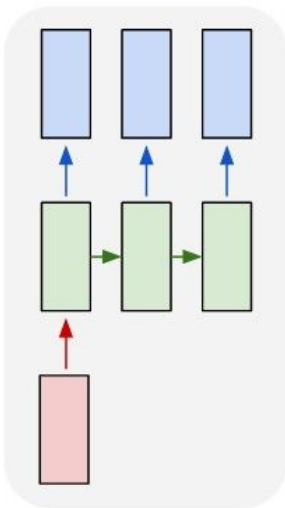
Para generar cada palabra, la red pesa los feature descriptors (**input**) de la imagen para identificar qué posiciones son importantes.

# Elegimos el tipo de red

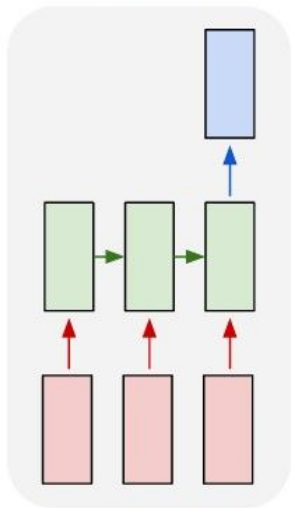
one to one



one to many

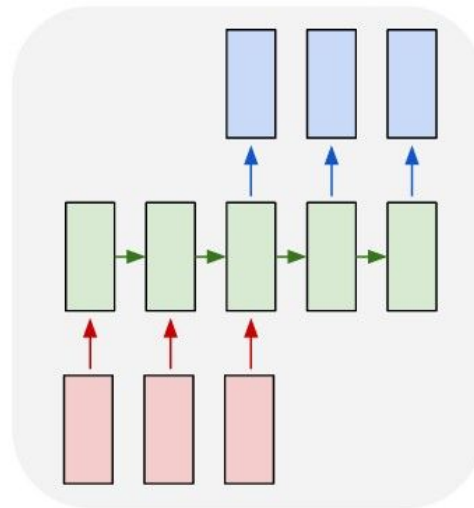


many to one



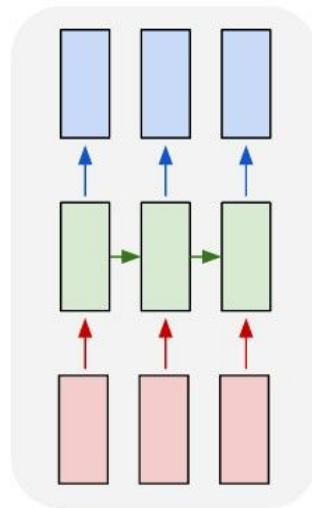
sequence  
classification

many to many



sequence  
to sequence

many to many



sequence  
labeling

PART II

# Mini Prototipo

# Resources

Dataset: [CoNLL NER task from Kaggle](#)

Código: Github repo

+ Notebook con las distintas redes

+ Visualización en JavaScript

Modelos entrenados

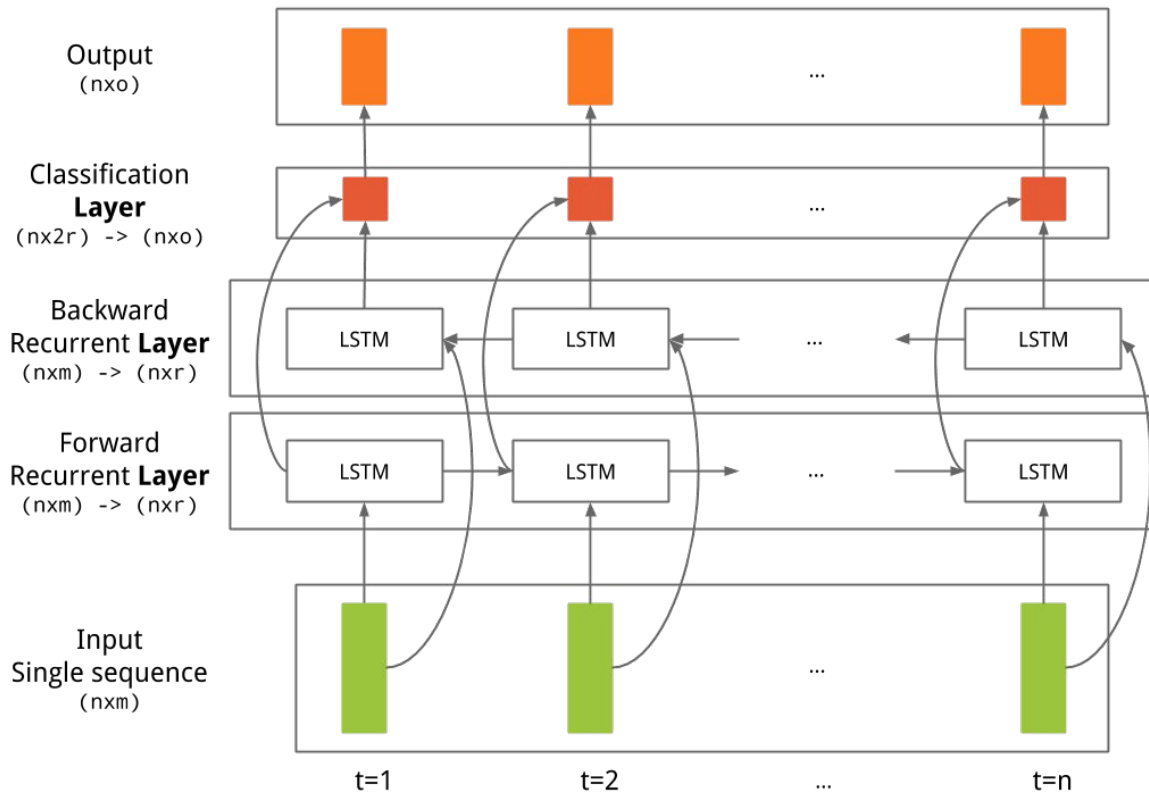
# Named Entity Recognition and Classification

Thousands of demonstrators have marched through London to protest the war in Iraq and demand the withdrawal of British troops from that country . Families of soldiers killed in the conflict joined the protesters who carried banners with such slogans as "Bush Number One Terrorist" and "Stop the Bombings".


GEO

GPE

PER



# General RNN for NERC



```
def add_embedding_layer(self, layers):
    layers = Embedding(
        input_dim=self.vocabulary_size,
        output_dim=self.max_sentence_length,
        input_length=self.max_sentence_length)(layers)
    return Dropout(0.1)(layers)

def add_recurrent_layer(self, layers):
    return Bidirectional(
        LSTM(units=100, return_sequences=True,
            recurrent_dropout=0.1))(layers)

def add_output_layer(self, layers):
    return TimeDistributed(
        Dense(self.n_labels, activation="softmax"))(layers)

def build(self):
    input = Input(shape=(max_sentence_length,))
    layers = self.add_embedding_layer(input)
    layers = self.add_recurrent_layer(layers)
    layers = self.add_output_layer(layers)

    self.model = Model(input, layers)
    self.model.compile(
        optimizer="adam", loss="categorical_crossentropy",
        metrics=["accuracy"])
```

Vanilla

LSTM

```
model.evaluate(X_test, y_test)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	995317
geo	0.84	0.85	0.85	9014
gpe	0.95	0.93	0.94	3177
per	0.88	0.85	0.86	6882
org	0.81	0.71	0.76	7301
tim	0.85	0.89	0.87	5196
art	0.39	0.18	0.25	144
nat	0.51	0.40	0.45	45
eve	0.42	0.47	0.45	104
avg / total	0.99	0.99	0.99	1027180

Vanilla

LSTM





Agreguemos atención





What and where  
to weight?



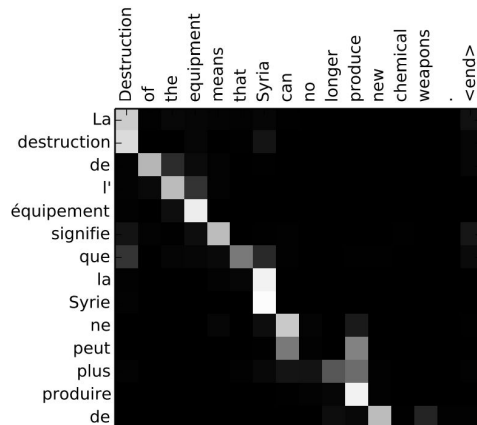
# Before Recurrence

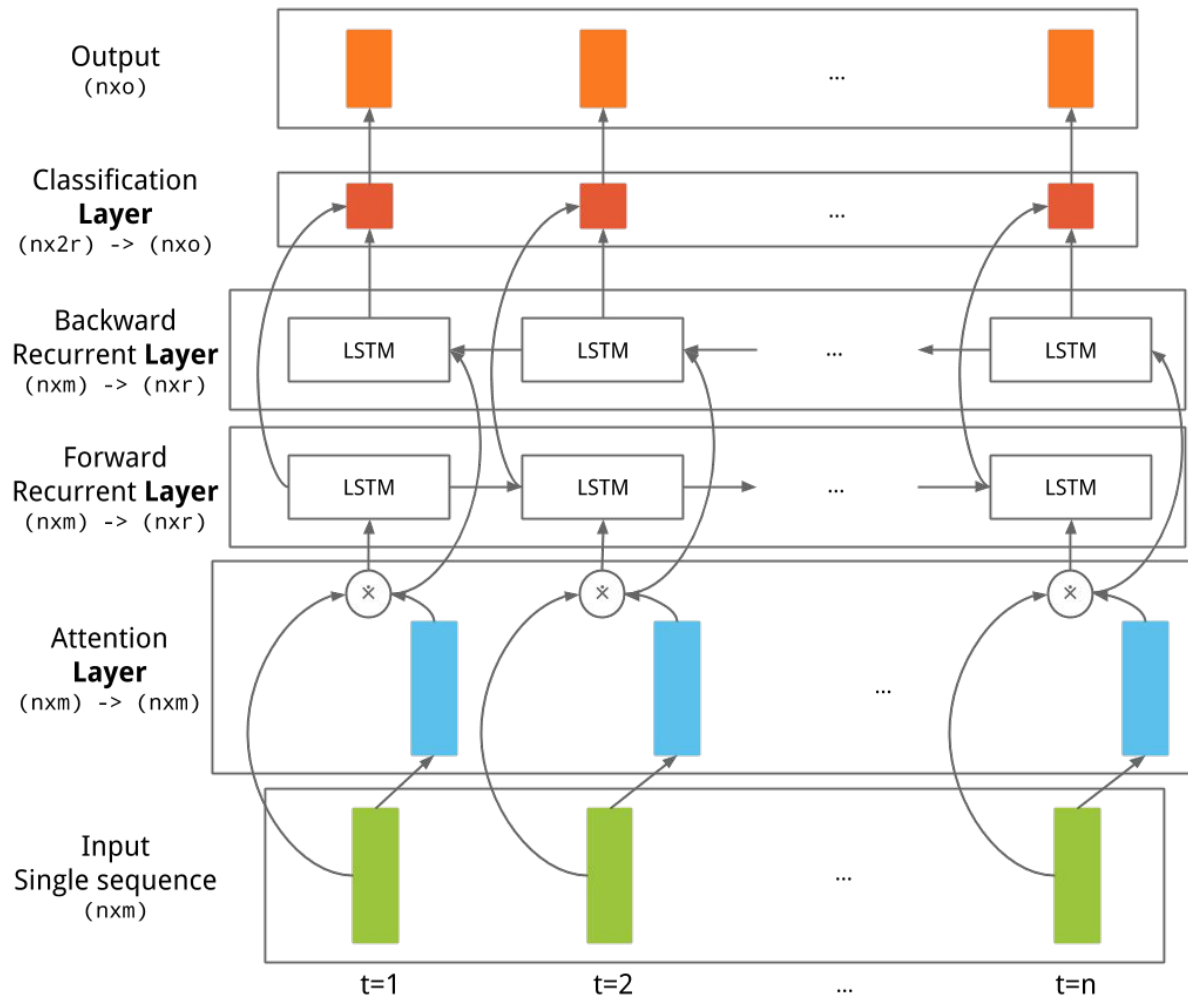
Podemos interpretar mejor cómo impacta el input en la clasificación. En general entendemos el espacio del input.

by ent423 ,ent261 correspondent updated 9:49 pm et ,thu march 19, 2015 ( ent261 ) a ent114 was killed in a parachute accident in ent45 ,ent85 ,near ent312 ,a ent119 official told ent261 on wednesday .he was identified thursday as special warfare operator 3rd class ent23 ,29 ,of ent187 , ent265 .` ent23 distinguished himself consistently

# After Recurrence

Podemos interpretar mejor qué información procesada es relevante para el cálculo del output





RNN + Att  
para NERC



```
class AttBiLSTM(BiLSTM):  
  
    def add_attention_block(self, layers):  
        """Apply an attention block to a partial model layers."""  
        return layers  
  
    def add_embedding_layer(self, layers):  
        layers = super(AttBiLSTM, self).add_embedding_layer(layers)  
        return self.add_attention_block(layers)
```


Definimos una nueva clase AttBiLSTM

y sobreescribimos la capa de


embeddings agregando un

post-proceso de atención

Soon to be  
  
attention  
  
LSTM



How to calculate the  
attention score?



# Inspiración: Philippe Remy



Para el problema de sequence labeling, el repositorio [Keras Attention Mechanism](#) de Philippe Remy implementa una solución muy básica

# Soft attention



Vector de pesos es **real**

La función es **derivable**

No funciona tan bien con secuencias  
largas [1]

# Hard attention



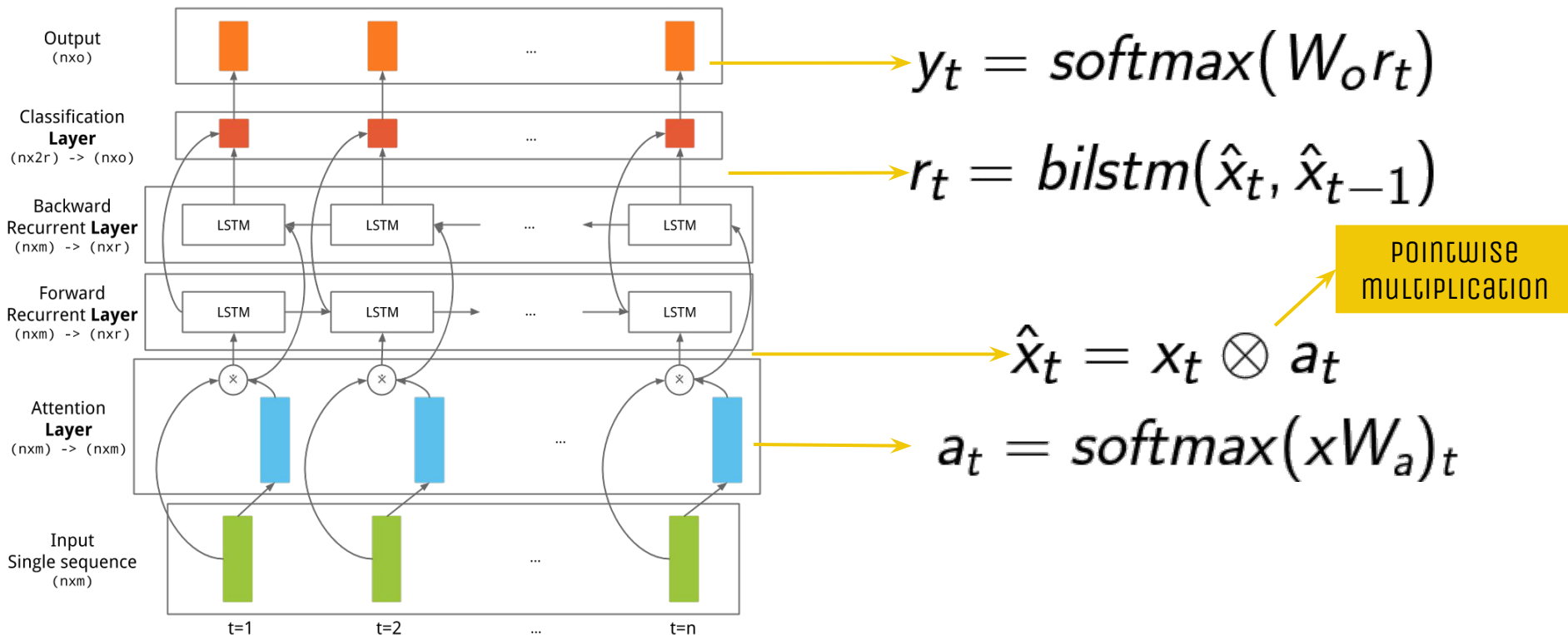
Vector de pesos es **binario**

La función **no es derivable**

Costoso de optimizar  
Útil para secuencias largas



# Formal definition



```
class AttBiLSTM(BiLSTM):
```

```
def add_attention_block(self, layers):  
    """Apply an attention block to a partial model layers."""  
    feature_vector_size = K.int_shape(layers)[-1]  
    att_layer = Dense(feature_vector_size, activation='softmax',  
                      name='attention_matrix_score')(layers)  
    layers = merge([att_layer, layers], mode='mul')  
    return layers  
  
def add_embedding_layer(self, layers):  
    layers = super(AttBiLSTM, self).add_embedding_layer(layers)  
    return self.add_attention_block(layers)
```

¡La atención es sólo una capa densa!

Calculamos un peso para cada **feature**

en cada **timestep**

# First attempt


## attention

## LSTM


```
class AttBiLSTM(BiLSTM):
```

```
def add_attention_block(self, layers):  
    """Apply an attention block to a partial model layers."""  
    feature_vector_size = K.int_shape(layers)[-1]  
    att_layer = Dense(feature_vector_size, activation='softmax',  
                      name='attention_matrix_score')(layers)  
    # Calculate a single score for each timestep  
    att_layer = Lambda(lambda x: K.mean(x, axis=1),  
                      name='attention_vector_score')(att_layer)  
    # Reshape to obtain the same shape as input  
    att_layer = Permute((2, 1))(  
        RepeatVector(feature_vector_size)(att_layer))  
    layers = merge([att_layer, layers], mode='mul')  
    return layers  
  
def add_embedding_layer(self, layers):  
    layers = super(AttBiLSTM, self).add_embedding_layer(layers)  
    return self.add_attention_block(layers)
```

Calculamos un peso para cada **timestep**,  
tomando el promedio de los pesos.




Finally  
attention  
LSTM



	precision	recall	f1-score	support
0	1.00	1.00	1.00	994779
geo	0.82	0.86	0.84	9071
gpe	0.96	0.89	0.92	3350
per	0.90	0.79	0.84	7014
org	0.81	0.69	0.75	7410
tim	0.87	0.86	0.87	5236
art	0.00	0.00	0.00	134
nat	0.00	0.00	0.00	55
eve	0.24	0.03	0.05	130
prev-prev-lemma	0.00	0.00	0.00	1
avg / total	0.99	0.99	0.99	1027180

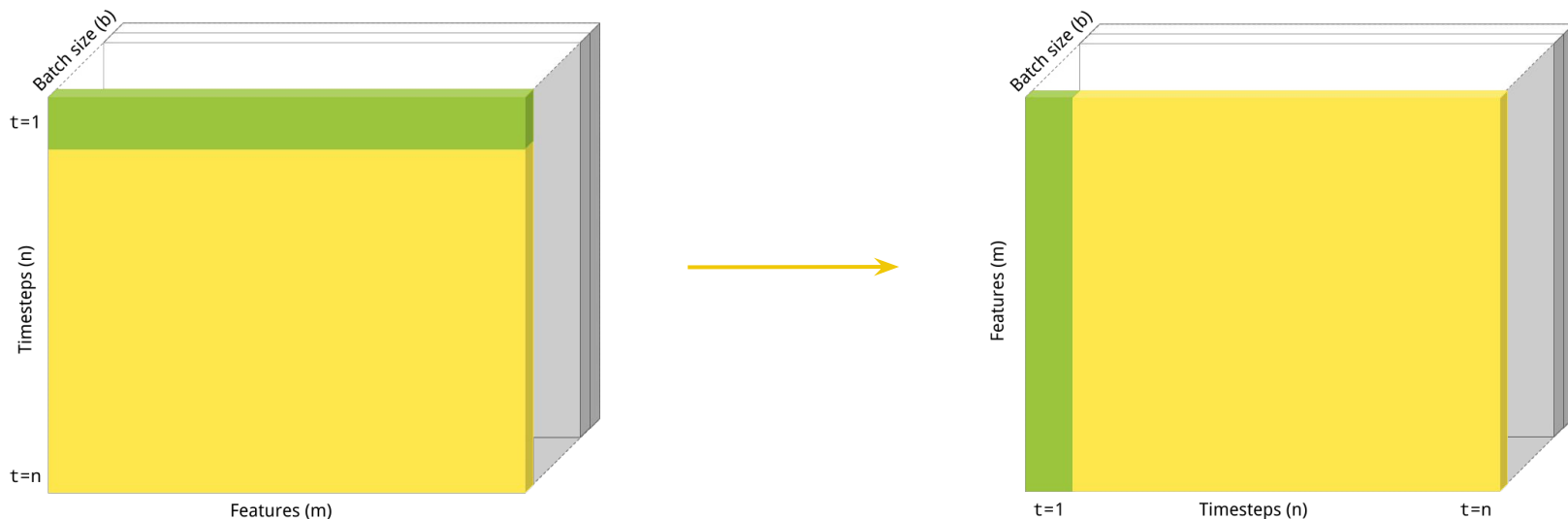
Finally  
attention  
LSTM



Cuando aplicamos la capa densa, cada attention score se calcula usando los valores de los **otros features** en el **mismo timestep**.

# Another solution

El repositorio [Keras Attention Mechanism](#) de Philippe Remy también implementa una solución muy básica, permutando las últimas dimensiones del input



```
class AttBiLSTM2(BiLSTM):
```

```
def add_attention_block(self, layers):  
    """Apply an attention block to a partial model layers."""  
    timesteps = K.int_shape(layers)[-2]  
    att_layer = Permute((2, 1))(layers)  
    att_layer = Dense(timesteps, activation='softmax', |  
                      name='attention_matrix_score')(att_layer)  
    # Calculate a single score for each timestep  
    att_layer = Lambda(lambda x: K.mean(x, axis=1),  
                      name='attention_vector_score')(att_layer)  
    # Reshape to obtain the same shape as input  
    att_layer = Permute((2, 1))(RepeatVector(timesteps)(att_layer))  
    layers = merge([att_layer, layers], mode='mul')  
    return layers  
  
def add_embedding_layer(self, layers):  
    layers = super(AttBiLSTM2, self).add_embedding_layer(layers)  
    return self.add_attention_block(layers)
```



Philippe Remy

attention

LSTM

Calculamos un peso para cada **timestep**,  
tomando el promedio de los pesos.

```
model.evaluate(X_test, y_test)
```


	precision	recall	f1-score	support
0	1.00	1.00	1.00	994779
geo	0.82	0.87	0.85	9071
gpe	0.96	0.89	0.92	3350
per	0.91	0.79	0.85	7014
org	0.84	0.68	0.75	7410
tim	0.91	0.85	0.88	5236
art	0.00	0.00	0.00	134
nat	0.00	0.00	0.00	55
eve	0.50	0.01	0.02	130
prev-prev-lemma	0.00	0.00	0.00	1
avg / total	0.99	0.99	0.99	1027180

Philippe Remy

attention

LSTM





Cuando aplicamos la capa densa, cada attention score se calcula usando el valor del **mismo feature** en **otros timesteps**.

## Time-wise

$$a_t = \textit{softmax}(xW_a)_t$$

La atención se basa solamente en la  
información de la instancia


Funciona con cualquier input

## Feature-wise

$$a_t = \textit{softmax}(x^T W_a)_t$$

La atención se basa en el promedio  
de la atención por feature

Necesitamos saber el tamaño de las  
secuencias de antemano



Si aplicamos una función de softmax a los attention scores, los transformamos en una distribución de probabilidad en  $(0, 1)$

**Suavizamos el valor de muchas neuronas**

## Without softmax - Model 1

```
model.evaluate(X_test, y_test)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	994779
geo	0.85	0.83	0.84	9071
gpe	0.96	0.91	0.93	3350
per	0.90	0.83	0.86	7014
org	0.73	0.75	0.74	7410
tim	0.90	0.86	0.88	5236
art	0.39	0.21	0.27	134
nat	0.47	0.47	0.47	55
eve	0.59	0.44	0.50	130
prev-prev-lemma	0.00	0.00	0.00	1
avg / total	0.99	0.99	0.99	1027180

## Without softmax - Model 2

	precision	recall	f1-score	support
0	1.00	1.00	1.00	994779
geo	0.85	0.83	0.84	9071
gpe	0.96	0.91	0.93	3350
per	0.90	0.83	0.86	7014
org	0.73	0.75	0.74	7410
tim	0.90	0.86	0.88	5236
art	0.39	0.21	0.27	134
nat	0.47	0.47	0.47	55
eve	0.59	0.44	0.50	130
prev-prev-lemma	0.00	0.00	0.00	1
avg / total	0.99	0.99	0.99	1027180



# Visualizing attention



## AttBiLSTM with softmax activation

Among those freed earlier this week were well-known dissident writer and poet Raul Rivero , opposition politician Osvaldo Alfonso Valdes and economist and journalist Oscar Espinosa Chepe .

Venezuela 's state-owned oil company says it is beginning to explore for oil in Cuban waters as part of a joint venture with the island 's state-owned Cubapetroleo .

## AttBiLSTM2 with softmax activation

Among those freed earlier this week were well-known dissident writer and poet Raul Rivero , opposition politician Osvaldo Alfonso Valdes and economist and journalist Oscar Espinosa Chepe .

Venezuela 's state-owned oil company says it is beginning to explore for oil in Cuban waters as part of a joint venture with the island 's state-owned Cubapetroleo .

## AttBiLSTM with linear activation

Among those freed earlier this week were well-known dissident writer and poet Raul Rivero , opposition politician Osvaldo Alfonso Valdes and economist and journalist Oscar Espinosa Chepe .

Venezuela 's state-owned oil company says it is beginning to explore for oil in Cuban waters as part of a joint venture with the island 's state-owned Cubapetroleo .

## AttBiLSTM2 with linear activation

Among those freed earlier this week were well-known dissident writer and poet Raul Rivero , opposition politician Osvaldo Alfonso Valdes and economist and journalist Oscar Espinosa Chepe .


Venezuela 's state-owned oil company says it is beginning to explore for oil in Cuban waters as part of a joint venture with the island 's state-owned Cubapetroleo .



# Take-home questions



**¿Interpretabilidad  
o performance?**



**¿Time-wise o  
feature-wise?**



**¿Aplicamos  
softmax o no?**

# Preguntas



mteruel@  
unc.edu.ar



[github.com/  
mit0110](https://github.com/mit0110)



[linkedin.com/in/  
milagro-teruel/](https://linkedin.com/in/milagro-teruel/)

# Referencias

- [1] Reinforced Self-Attention Network: a Hybrid of Hard and Soft Attention for Sequence Modeling
- [2] [Attention and memory in deep learning and nlp. Wildml](#)
- [3] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention