(CNN|BiLSTM)
+Word Embedding
+BiLSTM
+CRF

¿Cuánto cuesta llegar al estado del arte?

Hola, soy Milagro Teruel

Estudiante de doctorado en FaMAF

+ Representation Learning (embeddings)

+ Educational Data Mining

Argument mining

+ INRIA Sophia Antipolis

UNC Universidad Nacional de Córdoba

inventors for the digital world

Para ver hoy

+ Algunas tareas de sequence labeling

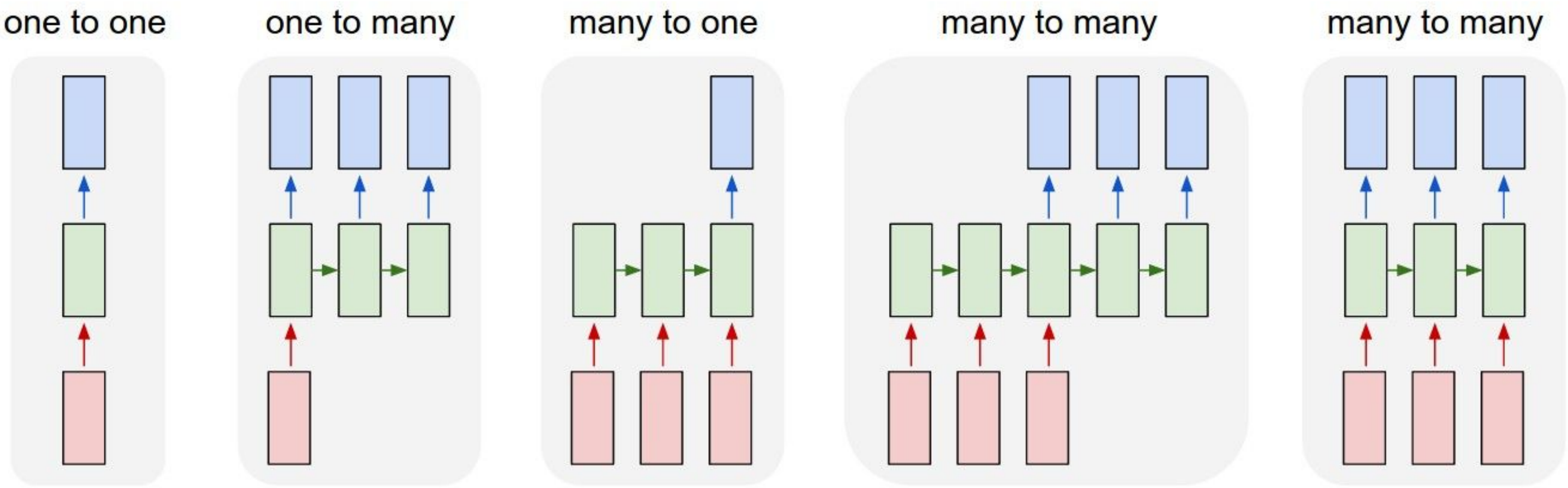
+ Red del estado del arte

(BiLSTM|CNN) BiLSTM CRF

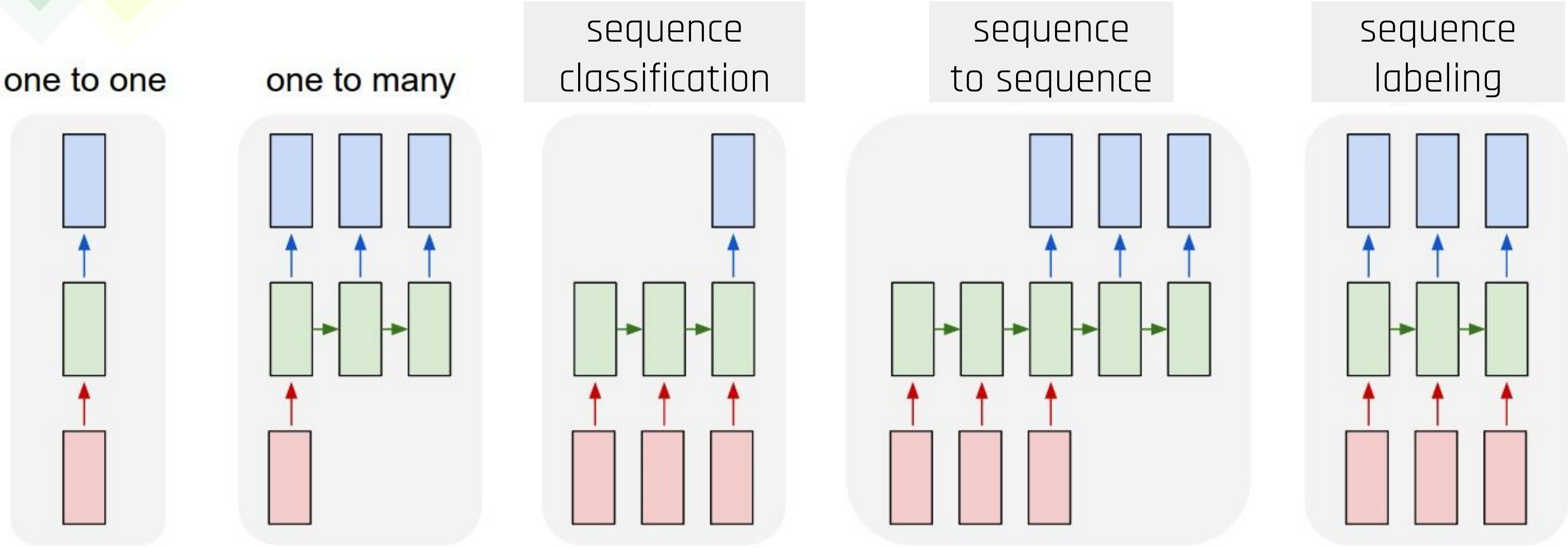
+ Cómo implementamos cada parte

1. Problemas

Tipos de problemas



Tipos de problemas



Sequence Labeling

+ PoS tagging

+ Text segmentation (chuncking)

+ Named Entity Recognition and Classification

+ Argument mining

Argument Mining

+ Reconocer estructuras de argumentación

+ Aplicado en juicios de la Corte Europea de Derechos Humanos (ECHR)

+ La semántica del problema es profunda

+ Las dependencias son de muy largo alcance

+ Muy pocos datos. Dataset anotado

CASE OF ALKASI VS TURKEY

[...]

The Court notes that the use as evidence [...] of statements given by the accused to the police [...] may amount to a violation of Article 6 § 1 of the Convention (see Salduz v. Turkey [GC], no. 36391/02, §§ 56-62, ECHR 2008). [...] Thus, the facts of the case seem to indicate that the statements given by the applicant to the police without the assistance of a lawyer were relied on by the labour court, [...]

[...]

Accordingly, there has been a violation of Article 6 § 2 of the Convention.

Premises are reasons given by the author to support or attack the claims. They are factual (not controversial).

Claims are controversial statements. Their acceptance depends on the premises that support or attack them.

NERC

+ Identificar las Entidades Nombradas en un texto, e.j. Persona, Lugar, Organización, Fecha, etc.

+ Existe información codificada en la estructura de la palabra

+ El contexto en el que se expresa una entidad es mucho menor

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

2. Arquitectura

Un estudio completo

Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging, *Nils Reimers and Iryna Gurevych*. EMNLP, 2017.

(extended version)

Selecting optimal parameters for a neural network architecture can often make the difference between mediocre and state-of-the-art performance

Hay para elegir

Arquitecturas:

+ BiLSTM-CRF (Huang et al., 2015)

+ CNN-BiLSTM-CRF (Ma and Hovy, 2016)

+ BiLSTM-BiLSTM-CRF architecture (Lample et al., 2016)

Hay para elegir

Pretrained word embeddings:

+ Komninos and Manandhar (2016)

+ Levy and Goldberg (2014)

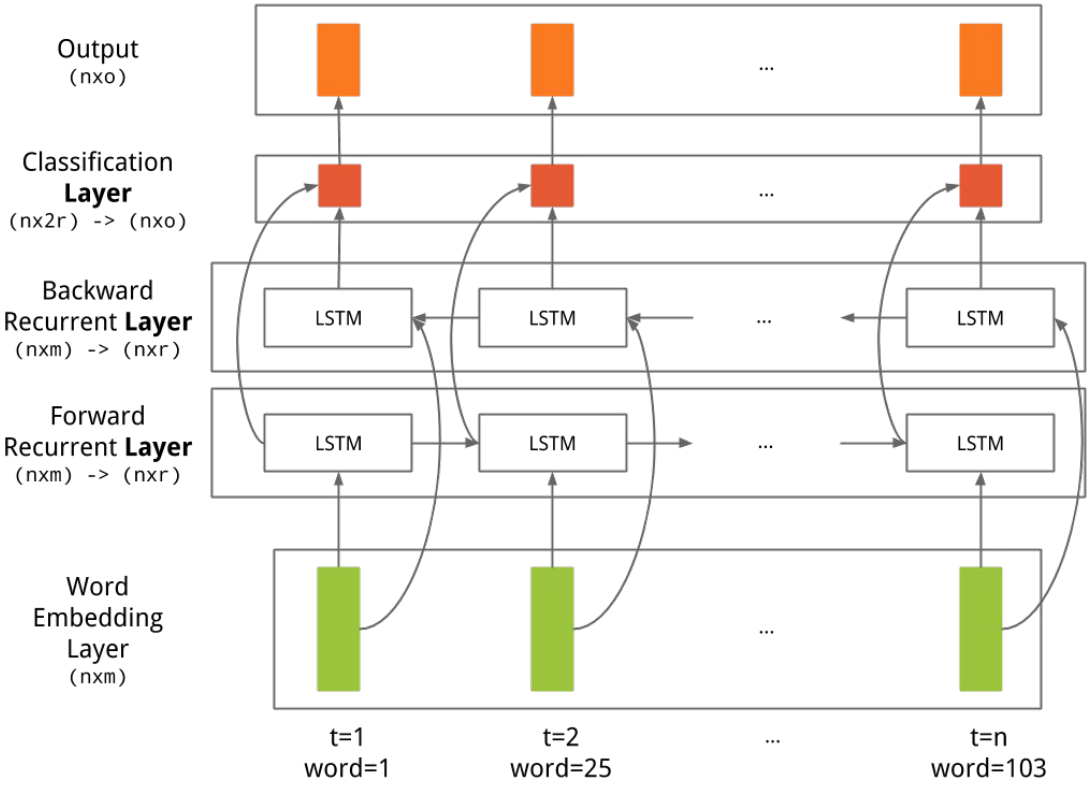
+ Mikolov et al (2013) [word2vec]

+ Pennington et al. (2014) [glove]

+ [Fasttext]

2.1 BiLSTM

Arquitectura básica



```python
def add_input_layer(self):
    return Input(shape=(None, ))

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

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

def add_output_layer(self, layers):
    layers = TimeDistributed(
        Dense(self.n_labels, activation='softmax'),
        name='dense_layer')(layers)
    return layers, 'categorical_crossentropy'

def build(self):
    inputs = self.add_input_layer()
    layers = self.add_embedding_layer(inputs)
    layers = self.add_recurrent_layer(layers)
    outputs, loss_function = self.add_output_layer(layers)

    self.model = Model(inputs=inputs, outputs=outputs)
    self.model.compile(optimizer='adam', loss=loss_function,
                       metrics=['accuracy'])
```

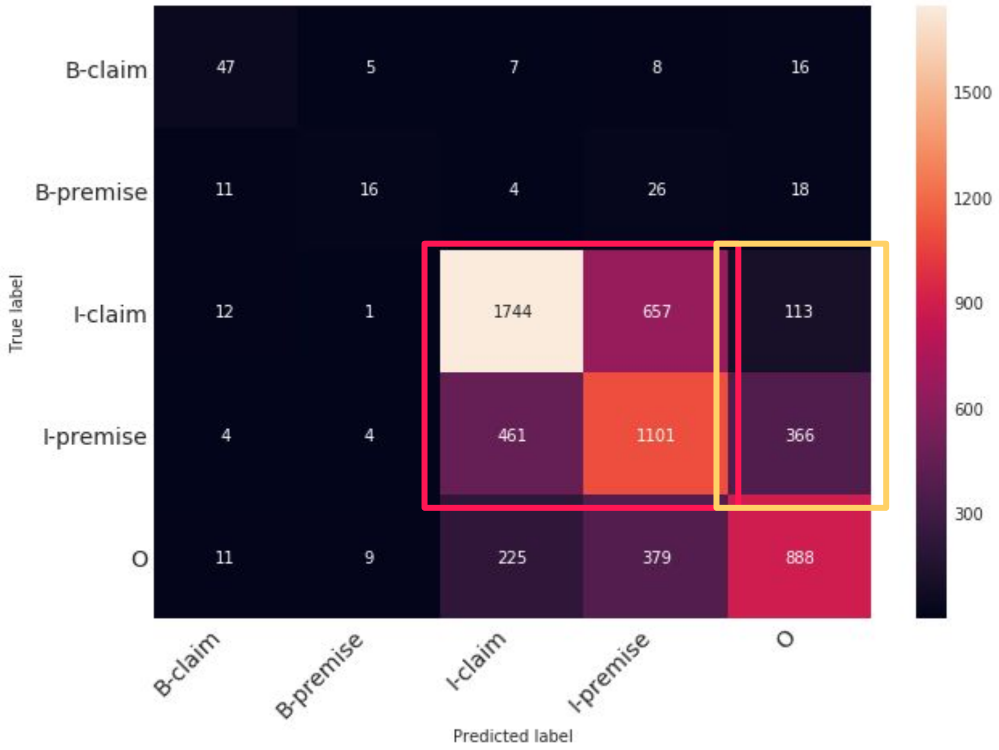
2.2 Word Embedding Pre-entrenados

¿Para qué sirven?

+ Pueden ser entrenados en un corpus mucho más grande
    + Aumentan el vocabulario
    + Capturan más información semántica
+ Aportan información a la red, posiblemente reduciendo el tiempo de entrenamiento y aumentando las posibilidades de convergencia

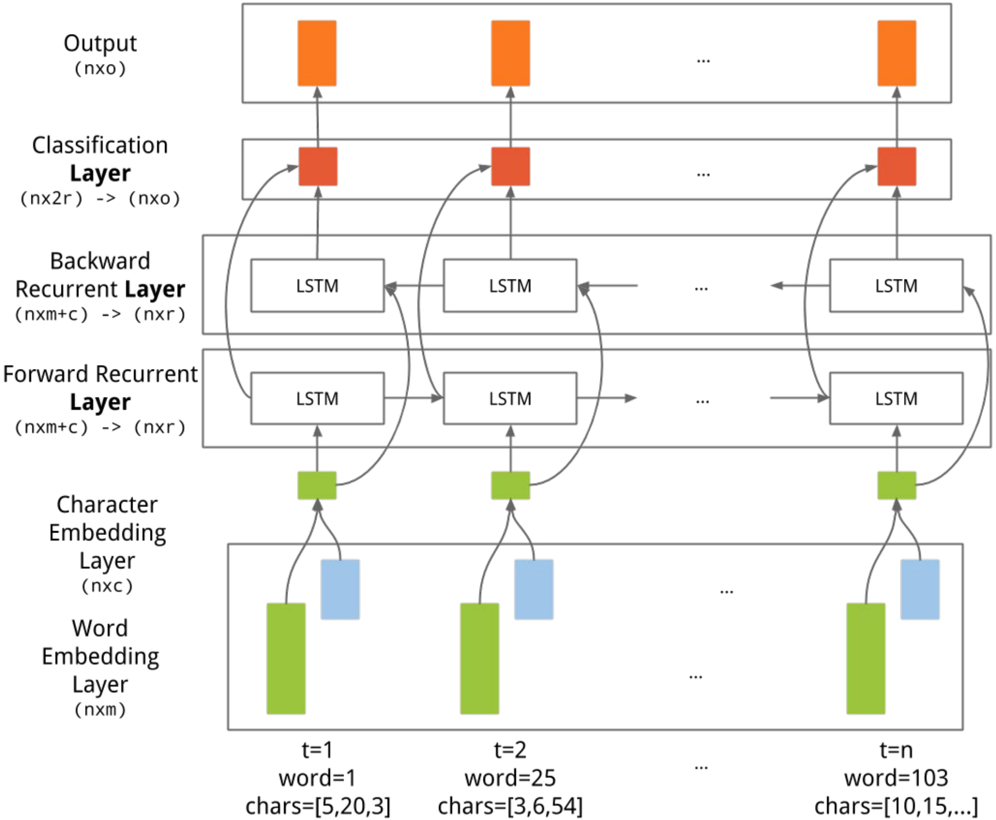
```python
def add_word_embeddings(self, embeddings):
    """Saves into the model a matrix with the original weights for
    the word embeddings.

    MUST BE CALLED BEFORE build, otherwise it has no effect.

    Args:
        embeddings: (numpy.ndarray) a 2-dimensional matrix with shape
            (vocabulary_size, embedding_size).
    """
    self.embeddings = embeddings
    self.embedding_size = embeddings.shape[1]  # Overwrite this value
    self.vocabulary_size = embeddings.shape[0]  # Overwrite this value

def add_embedding_layer(self, layers):
    if self.embeddings is not None:  # Add the pretrained embeddings
        layers = Embedding(
            input_dim=self.vocabulary_size, output_dim=self.embedding_size,
            weights=[self.embeddings],
            trainable=False, input_length=self.max_sentence_length)(layers)
    else:  # We use brand new embeddings
        layers = Embedding(
            input_dim=self.vocabulary_size, output_dim=self.embedding_size,
            input_length=self.max_sentence_length)(layers)
    return Dropout(0.1)(layers)
```

Resultados Base



2.3 CharEmbedding

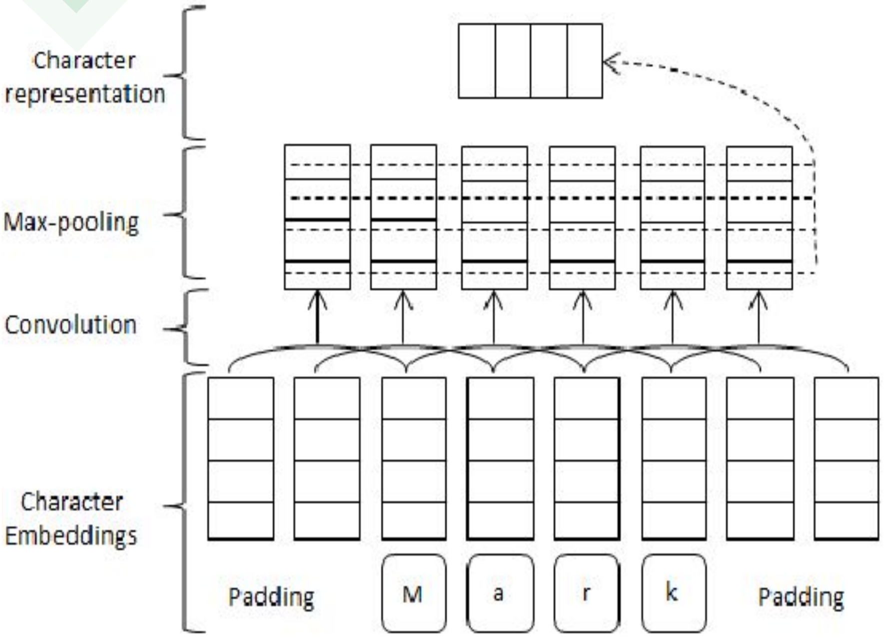
Arquitectura



¿Para qué sirven?

+ Permiten capturar información de palabras nunca antes vistas
+ Explotan características morfológicas
+ Corrigen ortografía, slang
+ Son independientes del lenguaje

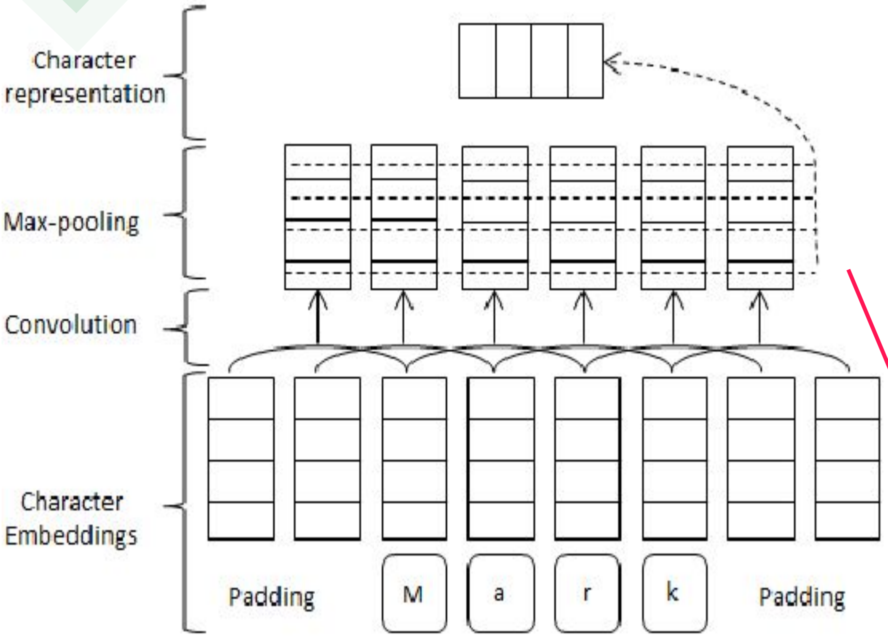
Propuesta 1: CNN



(a) CNN approach

```
def add_char_embedding_layer(self, char_input):
    """Add a convolution for the characters"""
    limit = numpy.sqrt(3.0/self.max_char_index)
    char_embedding_size = 15  # TODO define as a parameter
    # We initialize the char embeddings randomly,
    # including the UNK char in position 0.
    self.char_embeddings = numpy.random.uniform(
        -limit, limit, (self.max_char_index + 1,
                        char_embedding_size))

    # We need the TimeDistributed layer to embedd to every word
    chars_layer = TimeDistributed(Embedding(
        input_dim=self.char_embeddings.shape[0],
        output_dim=self.char_embeddings.shape[1],
        weights=[self.char_embeddings], trainable=True,
        mask_zero=True), name='char_embedding')(char_input)

    # Use CNNs for character embeddings from Ma and Hovy, 2016
    char_filter_size = 5  # TODO define this as a parameter
    char_filter_length = 5  # TODO define this as a parameter
    chars_layer = TimeDistributed(
        Conv1D(char_filter_size, char_filter_length, padding='same'),
        name="char_cnn")(chars_layer)
    chars_layer = TimeDistributed(GlobalMaxPooling1D(),
                                  name="char_pooling")(chars_layer)
    return chars_layer
```

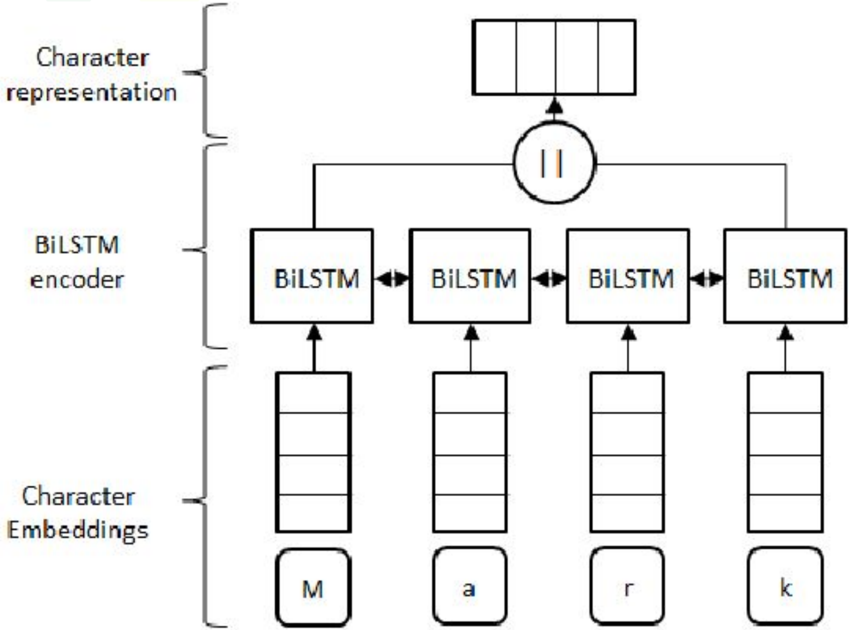
Propuesta 1: CNN



(a) CNN approach

```python
def add_char_embedding_layer(self, char_input):
    """Add a convolution for the characters"""
    limit = numpy.sqrt(3.0/self.max_char_index)
    char_embedding_size = 15  # TODO define as a parameter
    # We initialize the char embeddings randomly,
    # including the UNK char in position 0.
    self.char_embeddings = numpy.random.uniform(
        -limit, limit, (self.max_char_index + 1,
                        char_embedding_size))

    # We need the TimeDistributed layer to embedd to every word
    chars_layer = TimeDistributed(Embedding(
        input_dim=self.char_embeddings.shape[0],
        output_dim=self.char_embeddings.shape[1],
        weights=[self.char_embeddings], trainable=True,
        mask_zero=True), name='char_embedding')(char_input)

    # Use CNNs for character embeddings from Ma and Hovy, 2016
    char_filter_size = 5  # TODO define this as a parameter
    char_filter_length = 5  # TODO define this as a parameter
    chars_layer = TimeDistributed(
        Conv1D(char_filter_size, char_filter_length, padding='same'),
        name="char_cnn")(chars_layer)
    chars_layer = TimeDistributed(GlobalMaxPooling1D(),
                                  name="char_pooling")(chars_layer)
    return chars_layer
```

Propuesta 2: BiLSTM

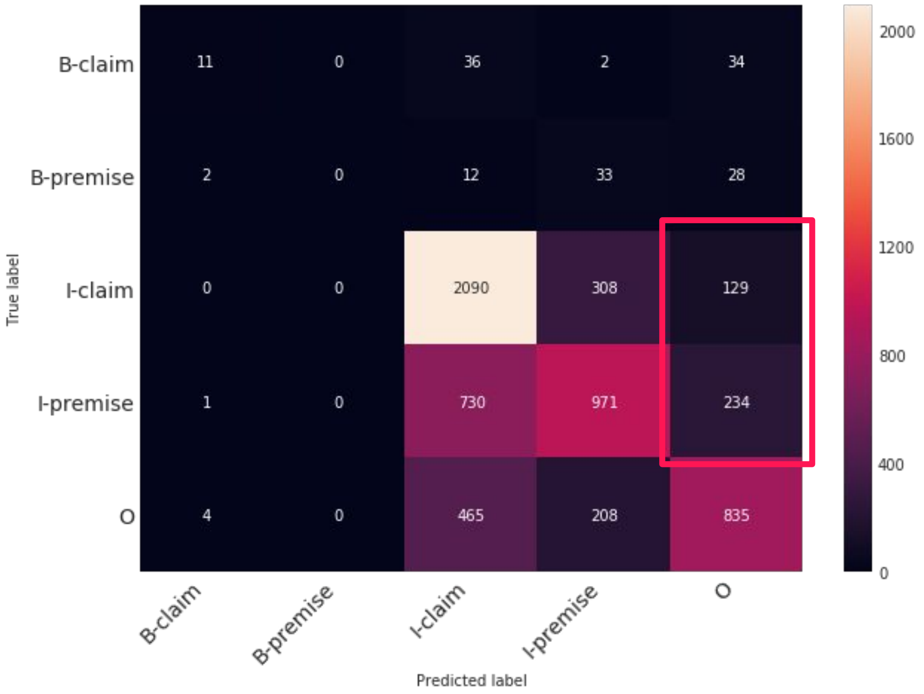


(b) BiLSTM approach

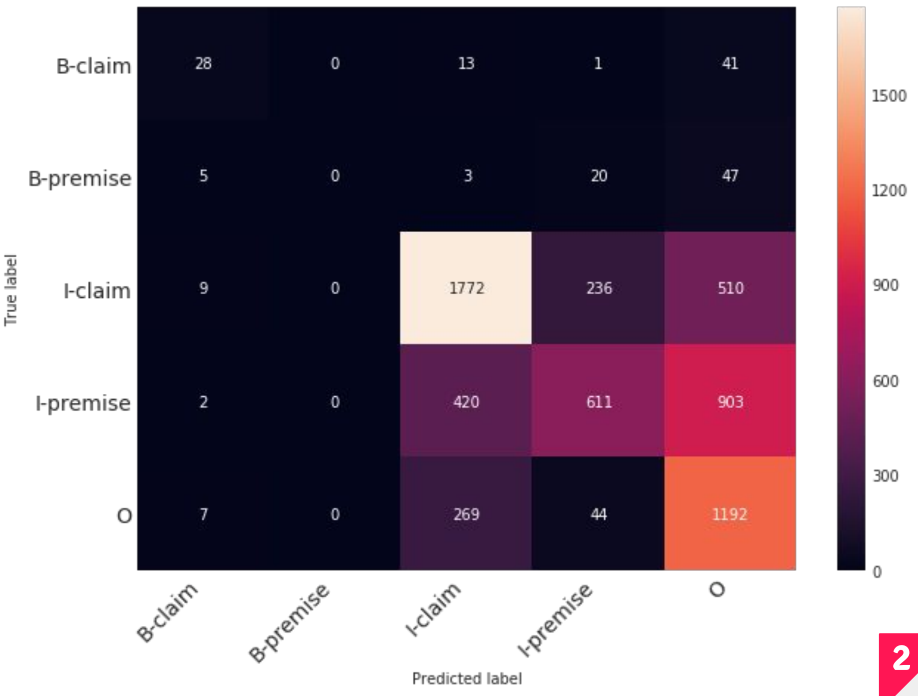
```
def add_char_embedding_layer(self, char_input):
    """Add a convolution for the characters"""
    limit = numpy.sqrt(3.0/self.max_char_index)
    char_embedding_size = 15  # TODO define as a parameter
    # We initialize the char embeddings randomly,
    # including the UNK char in position 0.
    self.char_embeddings = numpy.random.uniform(
        -limit, limit, (self.max_char_index + 1,
                        char_embedding_size))

    # We need the TimeDistributed layer to embedd to every word
    chars_layer = TimeDistributed(Embedding(
        input_dim=self.char_embeddings.shape[0],
        output_dim=self.char_embeddings.shape[1],
        weights=[self.char_embeddings], trainable=True,
        mask_zero=True), name='char_embedding')(char_input)

    # Use LSTM for char embeddings from Lample et al., 2016
    char_lstm_size = 10  # TODO define this as a parameter
    chars_layer = TimeDistributed(Bidirectional(
        LSTM(char_lstm_size, return_sequences=False)),
        name="char_lstm")(chars_layer)
    return chars_layer
```

+Char Emb Resultados

CNN

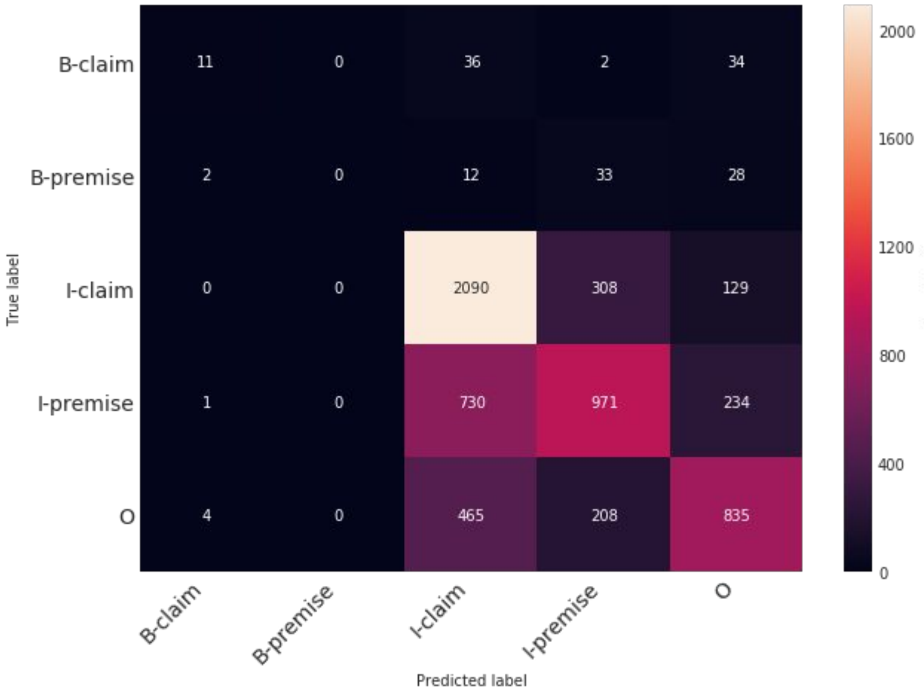


BiLSTM

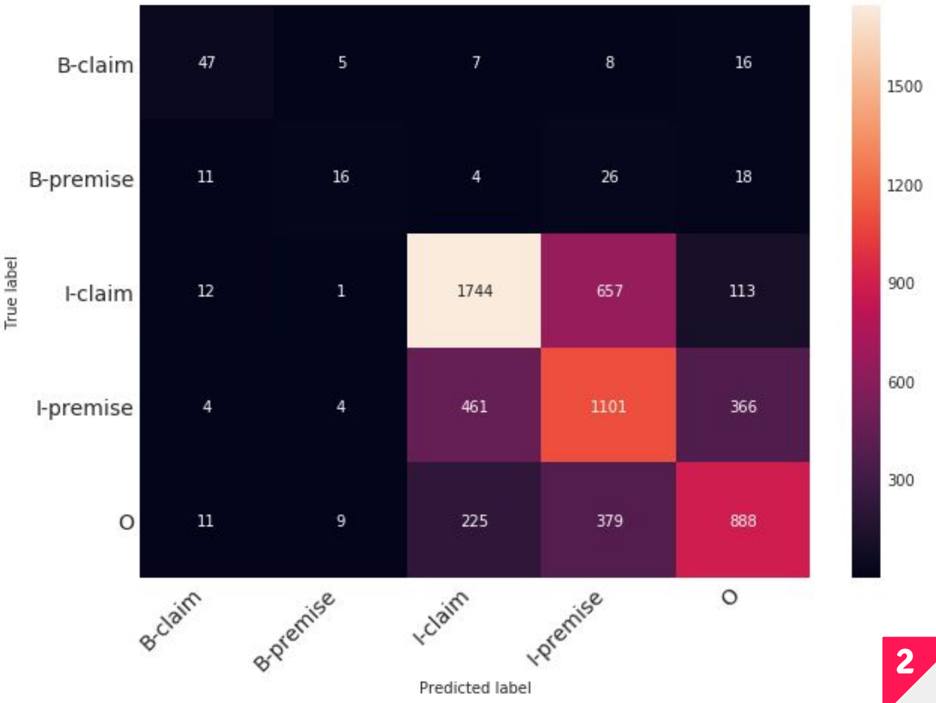


+Char Emb Resultados

CNN



No Char emb



Modelos radicalmente distintos,

misma performance:

¿Son realmente distintos para secuencias cortas?

2.4 CRF classification

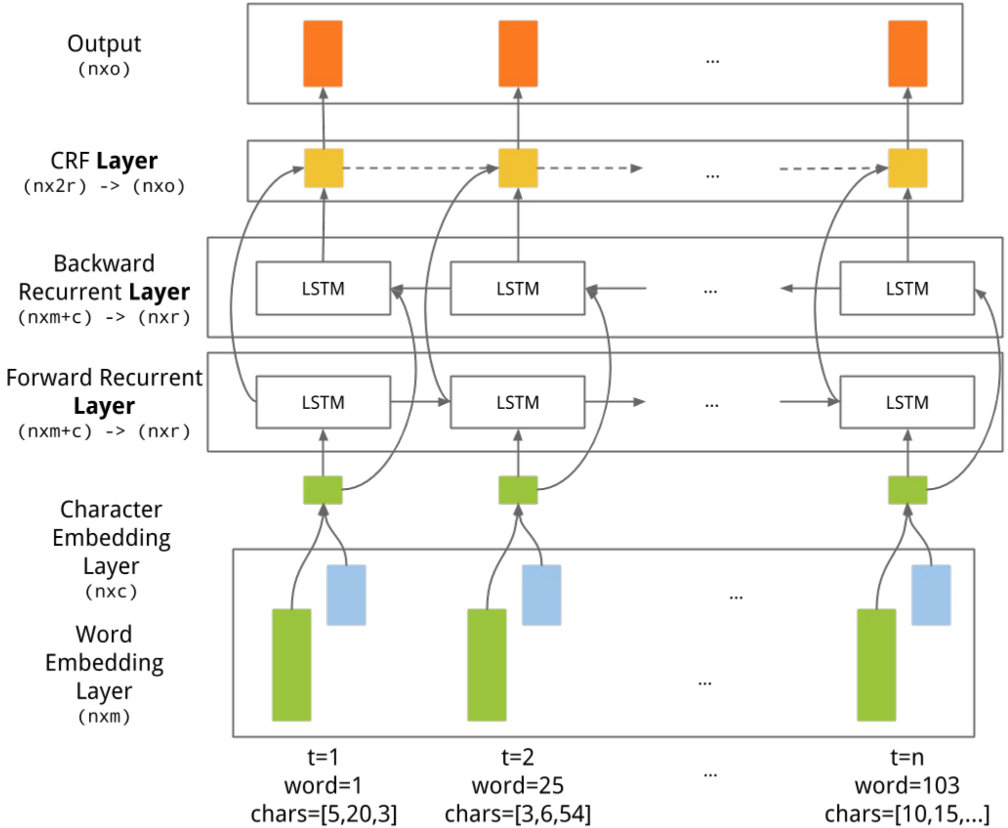
¿Para qué sirven?

+ Conditional Random Fields son clasificadores que tienen en cuenta el contexto cercano de la instancia

+ “Arreglan” las labels emitidas por la LSTM cuando son inconsistentes

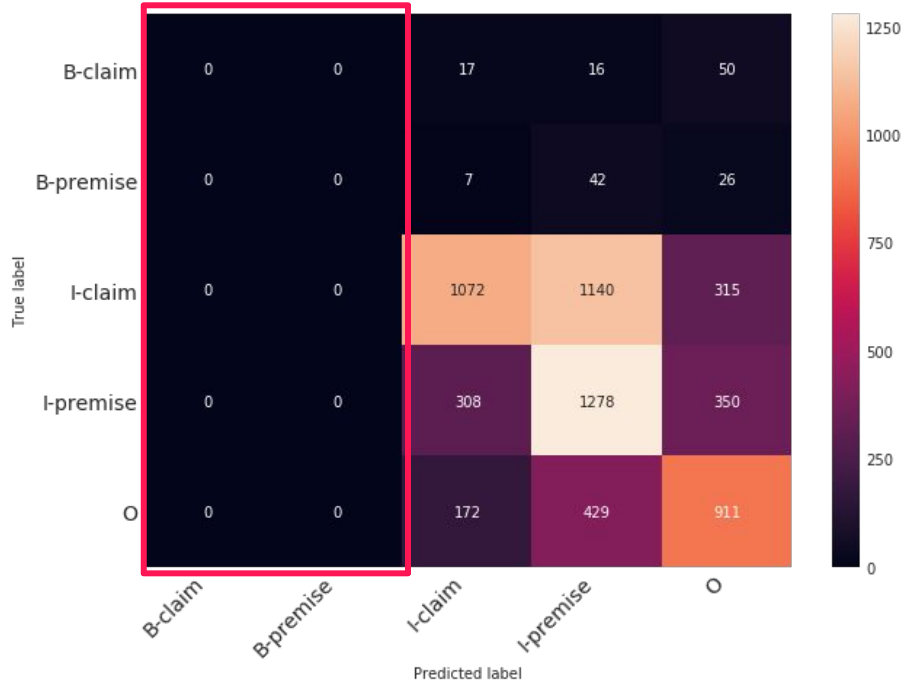
+ No hay implementaciones oficiales

Final architecture

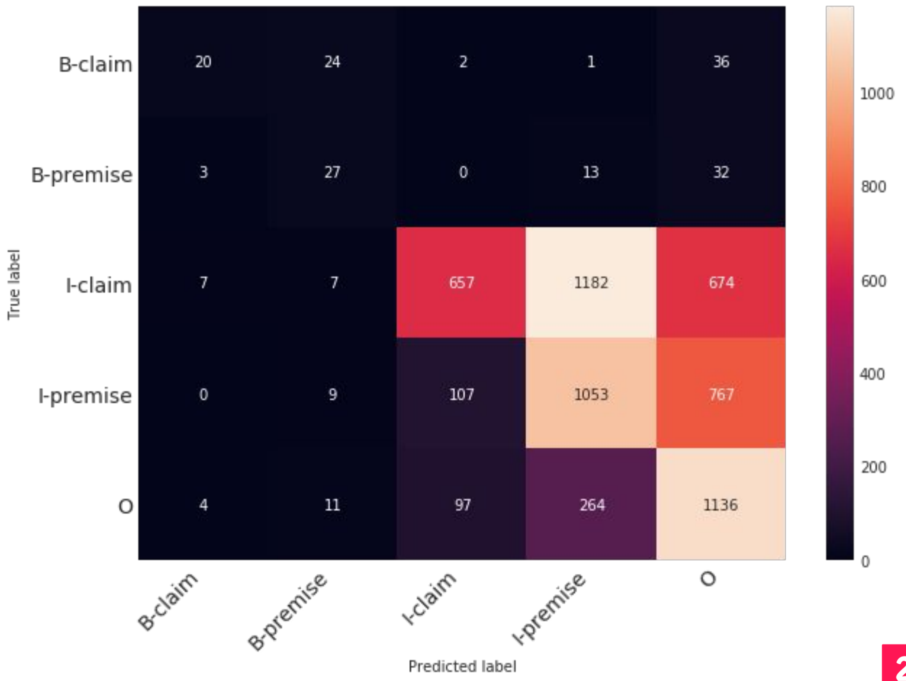


+CRF Resultados

Softmax



CRF



3. Training

Hiperparámetros

Architecture

Number of output layers
Type of output layer

Number of recurrent layers
Number of recurrent units
Type of cells

Training

Optimizer
Opt. hyperparameters
Batch size
Epochs
Early stopping

For each layer

Activations
Dropout
Regularization
Batch normalization
Grad clipping
Initializations

Input 1

Word embedding type
Word embedding size
Pretrained or not
Method hyperparameters

Input 2

Char embedding type
Char embedding size
Method hyperparameters

Take home messages

Criterio

Las redes tienen demasiadas configuraciones para tunearlas a mano

Innovación

Agregar otros tipos de embeddings no es costoso y puede mejorar la performance

Out.of.the.box

CRFs ayudan a corregir errores, pero son costosos de entrenar

¿Preguntas?

mteruel@unc.edu.ar

Referencias

+ Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. EMNLP

+ Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. CoRR

+ Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. CoRR,