

Master in Artificial Intelligence

Advanced Human Language Technologies

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner
- Classifier
- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables

Session 5 - NERC using neural networks

Assignment

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables

Write a python program that parses all XML files in the folder given as argument and recognizes and classifies drug names.

The program must use a neural network approach.

```
$ python3 ./nn-NER.py data/Devel/
```

```
DDI-DrugBank.d278.s0|0-9|Enoxaparin|drug
```

```
DDI-DrugBank.d278.s0|93-108|pharmacokinetics|group
```

```
DDI-DrugBank.d278.s0|113-124|eptifibatide|drug
```

```
DDI-MedLine.d88.s0|15-30|chlordiazepoxide|drug
```

```
DDI-MedLine.d88.s0|33-43|amphetamine|drug
```

```
DDI-MedLine.d88.s0|49-55|cocaine|drug
```

```
DDI-MedLine.d88.s1|82-95|benzodiazepine|drug
```

```
...
```

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner
- Classifier
- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables

General Structure

The general structure is basically the same than for the traditional ML approach:

- B-I-O schema
- Two programs: one learner and one classifier.
- The learner loads the training (Train) and validation (Devel) data, formats/encodes it appropriately, and feeds it to the model, together with the ground truth.
- The classifier loads the test data, formats/encodes it in the same way that was used in training, and feeds it to the model to get a prediction.

In the case of NN, we don't need to extract features (though we **do need** proper input encoding)

Input Encoding

- The input/output layers of a NN are vectors of neurons, each set to 0/1.
- Modern deep learning libraries handle this in the form of *indexes* (i.e. just provided the *position* of active neurons, omitting zeros).
- For instance, in a LSTM, each input word in the sequence may be encoded as the concatenation of different vectors each containing information about some aspect of the word (form, lemma, PoS, suffix...)
- Each vector will have only one active neuron, indicated by its *index*. This input is usually fed to an embedding layer.
- Our learner will need to create and store *index* dictionaries to be able to map the code assigned to each word, label, or any other used piece of information. See class *Codemaps* below.

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner
- Classifier
- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner

- Classifier

- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure
Learner

Core task

Goals &
Deliverables

Learner - Main program

```
1 def learn(traindir, validationdir, modelname) :
2     '''
3     learns a NN model using traindir as training data, and validationdir
4     as validation data. Saves learnt model in a file named modelname
5     '''
6     # load train and validation data in a suitable form
7     traindata = Dataset(traindir)
8     valdata = Dataset(validationdir)
9
10    # create indexes from training data
11    max_len = 150
12    suf_len
13    codes = Codemaps(traindata, max_len, suf_len)
14
15    # build network
16    model = build_network(idx)
17
18    # encode datasets
19    Xtrain = codes.encode_words(traindata)
20    Ytrain = codes.encode_labels(traindata)
21    Xval = codes.encode_words(valdata)
22    Yval = codes.encode_labels(valdata)
23
24    # train model
25    model.fit(Xtrain, Ytrain, validation_data=(Xval,Yval), batch_size=32,
26              epochs=10, verbose=1)
27
28    # save model and indexes, for later use in prediction
29    model.save(modelname)
30    codes.save(modelname+'/codemaps.txt')
```

Neural
Networks
NERC

General
Structure

Detailed
Structure

Learner

Core task

Goals &
Deliverables

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner

- **Classifier**

- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure
Classifier

Core task

Goals &
Deliverables

Classifier - Main program

```
1 def predict(modelname, datadir, outfile) :
2     '''
3     Loads a NN model from file 'modelname' and uses it to extract drugs
4     in datadir. Saves results to 'outfile' in the appropriate format.
5     '''
6
7     # load model and associated encoding data
8     model = load_model(modelname)
9     codes = Codemaps(modelname+'/codemaps.txt')
10
11     # load and encode data to annotate
12     testdata = Dataset(datadir)
13     X = codes.encode_words(testdata)
14
15     # tag sentences in dataset
16     Y = model.predict(X)
17     # get most likely tag for each word
18     Y = [[codes.idx2labels(np.argmax(w) for w in s) for s in Y]
19
20     # extract entities and dump them to output file
21     output_entities(testdata, Y, outfile)
22     # evaluate using official evaluator.
23     evaluation(datadir,outfile)
```

Neural
Networks
NERC

General
Structure

Detailed
Structure

Classifier

Core task

Goals &
Deliverables

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner

- Classifier

- **Auxiliary classes**

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure

Auxiliary classes

Core task

Goals &
Deliverables

Auxiliary classes - Dataset

```
1 class Dataset:
2     ## constructor: parses all XML files in datadir, tokenizes
3     ## each sentence, and
4     ## stores a list of sentences, each of them as a sequence of
5     ## tokens (word, start, end, gold_label)
6     def __init__(self, datadir):
7
8     ## iterator to get all sentences in the data set
9     def sentences(self):
10
11     ## iterator to get ids for sentence in the data set
12     def sentence_ids(self):
13
14     ## get one sentence (list of tokens) given its id
15     def get_sentence(self, sid) :
16     , , ,
```

Neural
Networks
NERC

General
Structure

Detailed
Structure

Auxiliary classes

Core task

Goals &
Deliverables

Auxiliary classes - Codemaps

```
1 class Codemaps :
2     # Constructor: create code mapper either from training data, or
3     #               loading codemaps from given file.
4     #               If 'data' is a Dataset, and lengths are not None,
5     #               create maps from given data.
6     #               If data is a string (file name), load maps from file.
7     def __init__(self, data, maxlen=None, suflen=None)
8     # Save created codemaps in file named 'name'
9     def save(self, name)
10    # Convert a Dataset into lists of word codes and suffix codes
11    # Adds padding and unknown word codes.
12    def encode_words(self, data)
13    # Convert the gold labels in given Dataset into a list of label codes.
14    # Adds padding
15    def encode_labels(self, data)
16    # get word index size
17    def get_n_words(self)
18    # get suf index size
19    def get_n_sufs(self)
20    # get label index size
21    def get_n_labels(self)
22    # get index for given word
23    def word2idx(self, w)
24    # get index for given suffix
25    def suff2idx(self, s)
26    # get index for given label
27    def label2idx(self, l)
28    # get label name for given index
29    def idx2label(self, i)
```

Neural
Networks
NERC

General
Structure

Detailed
Structure

Auxiliary classes

Core task

Goals &
Deliverables

Required functions - build_network

```
1 def build_network(codes) :
2
3     # sizes
4     n_words = codes.get_n_words()
5     n_sufs = codes.get_n_sufs()
6     max_len = codes.maxlen
7
8     inptW = Input(shape=(max_len,)) # word input layer & embeddings
9     embW = Embedding(input_dim=n_words, output_dim=100,
10                      input_length=max_len, mask_zero=True)(inptW)
11
12     inptS = Input(shape=(max_len,)) # suf input layer & embeddings
13     embS = Embedding(input_dim=n_sufs, output_dim=50,
14                      input_length=max_len, mask_zero=True)(inptS)
15
16     dropW = Dropout(0.1)(embW)
17     dropS = Dropout(0.1)(embS)
18     drops = concatenate([dropW, dropS])
19
20     bilstm = Bidirectional(LSTM(units=200, return_sequences=True,
21                                recurrent_dropout=0.1))(drops)
22
23     out = TimeDistributed(Dense(n_labels, activation="softmax"))(bilstm)
24
25     model = Model([inptW,inptS], out)
26     model.compile(optimizer="adam",
27                  loss="sparse_categorical_crossentropy",
28                  metrics=["accuracy"])
29
30     return model
```

Neural
Networks
NERC

General
Structure

Detailed
Structure

Auxiliary classes

Core task

Goals &
Deliverables

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner
- Classifier
- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables

Build a good NN-based drug NERC

Strategy: Experiment with different architectures and possibilities.
Some elements you can play with:

- Embedding dimension
- Initializing word embeddings with available pretrained models
- Max length and suffix length values
- Number of LSTM units
- Used optimizer
- Number and kind of layers or activation functions
- Additional input layers (maybe with embeddings). **Attention:**
This will require extending class `Codemaps` to handle the codes of added input layers.
 - lowercased words
 - different length suffixes and/or prefixes
 - PoS tags
 - feature layer (with information about capitalization, dashes, presence in external resources, etc)

Build a good NN-based drug NERC

Warnings:

- Neural Network training uses randomization, so different runs of the same program will produce different results. For repeatable results, use a random seed.
- During training, Keras reports *accuracy* on training and validation sets. Those values are usually over 90%. However, this is due to the fact that most of the words have label “0” (non-drug). Accuracy values around 90% roughly correspond to F_1 values around 25%. To get a reasonable F_1 , validation set accuracy should reach about 97%.

To precisely evaluate how your model is doing, do not rely on reported accuracy: run the classifier on the development set and use the evaluator.

Outline

1 Neural Networks NERC

2 General Structure

3 Detailed Structure

- Learner
- Classifier
- Auxiliary classes

4 Core task

5 Goals & Deliverables

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables

Exercise Goals

What you should do:

- Work on your architecture and input vectors. It is the component of the process where you have most control.
- Experiment with different architectures and hyperparameters.
- Experiment with different input information
- Keep track of tried variants and parameter combinations.

What you should **NOT** do:

- Alter the suggested code structure (i.e. change only `build_network` and `Codemaps`).
- Produce an overfitted model: If performance on the test dataset is much lower than on devel dataset, you probably are overfitting your model.

Exercise Goals

Orientative results:

- A biLSTM with 2 input layers (word and suffix embeddings) is enough to get a macroaverage F1 about 55%.
- Adding input layers with lowercased words and additional features (capitalization, dashes, numbers, presence in external files, ...) raises the score over 70%

Results much lower than these orientative scores is an indication that you are doing something wrong or not elaborated enough.

Deliverables

- You'll be expected to produce a report on neural approaches to NER and DDI.
- By now, just keep track of the information you'll need later:
 - Experimented architectures/hyperparameters
 - Experimented input information
 - Performance results on devel corpus using different configurations
 - Performance results on test corpus using different configurations

Neural
Networks
NERC

General
Structure

Detailed
Structure

Core task

Goals &
Deliverables