Named Entity Recognition

... reviewing machine learning literature about named entity recognition

A Survey on Deep Learning for Named Entity Recognition

- Goal : provide a comprehensive review on existing deep learning techniques for NER
- Link : https://arxiv.org/abs/1812.09449 (03/2020)
- Code : -
- Credible source: IEEE Transactions on Knowledge and Data Engineering

Named Entity Recognition (NER)

- NER is the process of locating and classifying named entities in text into predefined entity categories (tags)
- there are two types generic NEs (person, location etc.) and domain-specific NEs (proteins, genes etc.)
- plays an essential role in a variety of natural language processing (NLP) applications

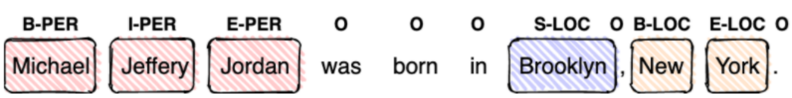
NER Dataset

corpus	url	source	number of tag
CoNLL03 (2003)	https://www.clips.uantwerpen.be/conll2003/ner	Reuters news - large portion of sports news	4 (person, locations, organisations, and miscellaneous)
OntoNotes (2007-2012)	https://catalog.ldc.upenn.edu/LDC2013T19	magazine, news, web, etc.	18 (person, location, product, event, language, time, date, etc.)

NER evaluation

- NER systems are usually evaluated by comparing their outputs against human annotations
- comparison can be quantified by either
    1. exact-match
        a. confusion matrix
        b. calculate precision, recall, and F-score
    2. relaxed match
        a. a correct type is credited if an entity is assigned its correct type regardless its boundaries as long as there is an overlap with ground truth boundaries
        b. not intuitive and make error analysis difficult, not widely used in recent studies

NE Tagging system - BIESO, BIO system



Model

	approaches	description

traditional	rule-based
	• example of systems
	  ◦ Brill rule inference approach for speech input
	  ◦ ProMiner in biomedical domain
	  ◦ LaSIE-II, NetOwl , Facile , SAR, FASTUS, and LTG
		• do not need annotated data as they rely on hand-crafted rules
		• rules can be designed based on domain-specific gazetteers, and syntactic-lexical patterns
		• works well when lexicon is exhaustive
		• due to domain-specific rules and incomplete dictionaries, high precision and low recall are often observed
		• cannot be transferred to other domains.

<table>
<tr>
<td></td>
<td>unsupervised learning

https://dl.acm.org/doi/abs/10.5555/1090483.1644538
https://www.researchgate.net/publication/28764761_Unsupervised_Named-Entity_Recognition_Generating_Gazetteers_and_Resolving_Ambiguity

</td>
<td>

rely on unsupervised algorithms without hand-labeled training examples
clustering-based NER systems extract named entities from the clustered groups based on context similarity

</td>
</tr>
<tr>
<td></td>
<td>feature-based supervised learning

Hidden Markov Models (HMM)
named IdentiFinder - first HMM-based NER system
Decision Trees
multilingual NER system - using C4.5 decision tree and AdaBoostM1 learning algorithm
Maximum Entropy Models
Maximum Entropy Named Entity (MENE)
Support Vector Machines (SVM)
Conditional Random Fields (CRF)

</td>
<td>

rely on supervised learning algorithms with careful feature engineering
a multi-class classification or sequence labelling task

</td>
</tr>
<tr>
<td>deep learning based</td>
<td colspan="2">deep learning

a field of machine learning that is composed of multiple processing layers to learn representations of data with multiple levels of abstraction
typical layers are artificial neural networks which consists of the forward pass and backward pass
the forward pass computes a weighted sum of their inputs from the previous layer and pass the result through a non-linear function
the backward pass is to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules via the chain rul
the key advantage

the capability of representation learning and the semantic composition empowered by both the vector representation and neural processing
this allows a machine to be fed with raw data and to automatically discover latent representations and processing needed for class

Why deep learning for NER?

the non-linear transformation generates non-linear mappings from input to output
compared with linear models, DL-based models are able to learn complex and intricate features from data via non-linear activation functions
saves significant effort on designing NER features
the traditional feature based approaches require considerable amount of engineering skill and domain expertise
can be trained in an end-to-end paradigm, by gradient descent
this property enables us to design possibly complex NER systems

</td>
</tr>
<tr>
<td></td>
<td>localist representation</td>
<td></td>
</tr>
</table>

- each processing unit can contribute to only one concept
- unit 1 only 'fires' for one concept ('small red car') and doesn't involve itself in representing anything else
- one hot encoding
    - have a vocabulary of n words and represent each word using a vector that is n bits long, in which all bits are zero except for one bit that is set to 1
    - vectors are very long as the number of words are getting larger high computational cost, high dimensionality
    - all words are equally similar from each other (not semantic)
- https://www.districtdatalabs.com/nlp-research-lab-part-1-distributed-representations

representation

Concept	Representation
Small Red Car	[1]
Not a Small Red Car	[0]

Concept	Representation
Small Red Car	[1 0]
Large Blue SUV	[0 1]

Concept	Representation
Small Red Car	[1 0 0]
Large Blue SUV	[0 1 0]
Large Red SUV	[0 0 1]

" Duct tape works
should be worsh

"duct" -->

"tape" -->

...

"magic" -->

...

"worshiped" -->

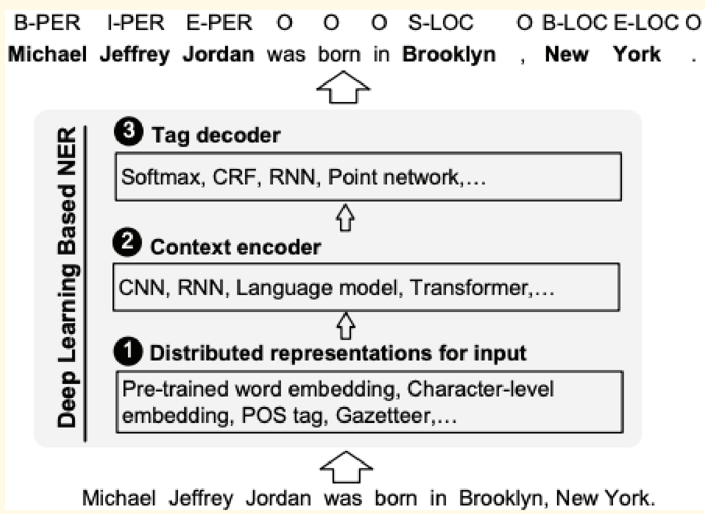
- a machine learning model can't directly see, hear, or sense input examples
- instead, you must create a representation of the data to provide the model with a useful vantage point into the data's key qualities
- in order to train a model, you must choose the set of features that best represent the data

distributed representation

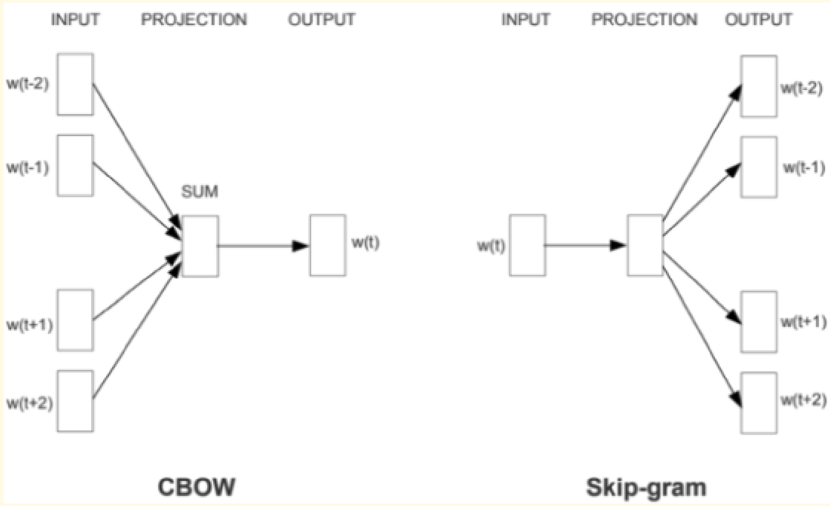
- representation of any single concept is distributed over many
- the unit values in the vectors are continuous values
- each processing unit contributes to any and all concepts
- the representations are dense (vs. localist representations which are sparse)
- concepts are no longer localised in one unit (hence the “distributed” designation)
- able to represent a very large number of concepts with only 4 processing units (as opposed to being limited by n units to n concepts)
- can learn new concepts without adding new units, all we need is a new configuration of values

Concept	Representation
Small Red Car	[0.555 0.761 0.243 0.812]
Large Blue SUV	[0.773 0.309 0.289 0.835]
Large Red SUV	[0.766 0.780 0.294 0.834]
Green Apple	[0.153 0.022 0.654 0.513]
Bumble Bee	[0.045 0.219 0.488 0.647]
Tall Building	[0.955 0.085 0.900 0.773]
Small Fish	[0.118 0.192 0.432 0.618]
Banana	[0.184 0.232 0.671 0.589]

	• most importantly, able to represent similarities better
	• word embeddings
	    • take a corpus of text, and figure out distributed vector representations of words that retain some level of semantic similarity between them
	    • can be used as inputs to additional models such as an SVM or recurrent neural network	

	distributed representation for input
	
	• represents words in real-valued dense vectors where each dimension represents a latent feature
	• automatically learned from text
	• captures semantic and syntactic properties of word
	• different types of distributed representations
	    • word-level representation
	    • character-level representation
		The architecture of DL-based NER
		
		
		
		1. distributed representation for input
		    • word- and character-level embec
		2. context encoder
		    • capture the context dependencie
		3. tag decoder
		    • predict tags for tokens in the inpu

- hybrid representation
  - additional information (e.g., gazetteers, lexical similarity, linguistic dependency and visual features) into the final representations of words, before feeding into context encoding layers
  - DL-based representation combined with feature-based approach
  - In the BiLSTM-CRF model, the extra features (i.e., gazetteers) boost tagging accuracy (“Bidirectional lstm-crf models for sequence tagging”)

word-level representation
 	
• pre-trained over large collections of text through unsupervised algorithms such as Continuous Bag-Of-Words (CBOW) and continuous skip-gram models
• CBOW and Skip-gram are architectures to learn the underlying word representations for each word by using neural networks
• Continuous Bag-Of-Words
  ◦ predicts a target word using the surrounding words
  ◦ uses continuous representations whose order is of no importance
  ◦ the sum of the background vectors is used
  ◦ pre-defined window size surrounding the target word defines the neighbouring terms that are taken into account
  ◦ several times faster to train than the skip-gram, slightly better accuracy for the frequent words
	

<table>
<tr>
<td>

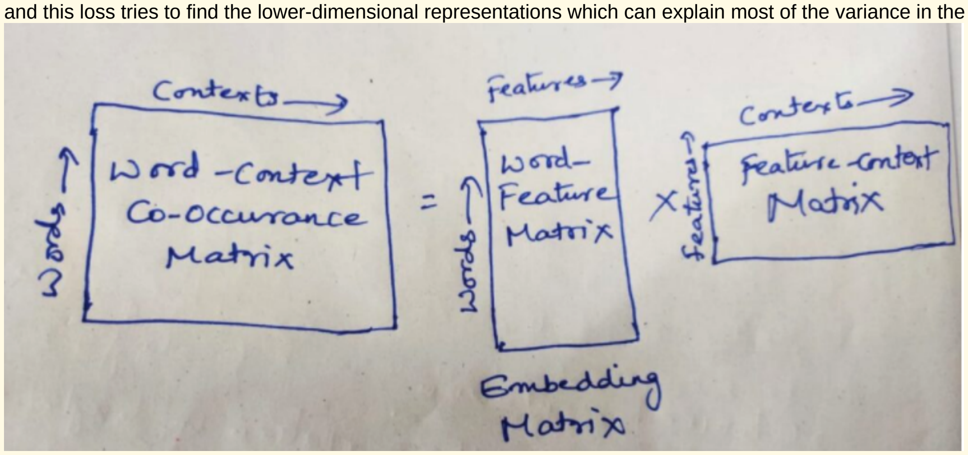
- Skip-gram
    - predicts the surrounding words using the target word
    - sums the log probabilities of the surrounding words to the left to the right of the target word
    - works well with small amount of the training data, represents well even rare words or phrases
- https://aylien.com/blog/overview-word-embeddings-history-word2vec-cbow-glove

</td>
<td></td>
</tr>
<tr>
<td>

commonly used word embeddings

- Google word2Vec
    1. treats each word in corpus like an atomic entity and generates a vector for each word
    2. based on training a shallow feedforward neural network

</td>
<td>

- how is the word2vec model trained CBOW, Skip-gram
- how is the GloVe model trained
    - first constructs a large matrix of (words x context) co-occurrence information
    - for each “word” (the rows), you count how frequently we see this word in some “context” (the columns) in a large cor
    - https://medium.com/ai-society/jkljlj-7d6e699895c4
    - then factorise this matrix to yield a lower-dimensional (word x features) matrix, where each row now yields a vector r
    - in general, this is done by minimising a “reconstruction loss”
    - and this loss tries to find the lower-dimensional representations which can explain most of the variance in the high-di
    - 

</td>
</tr>
</table>

- Stadford GloVe
  1. treats words as the smallest unit to train on (like word2vec)
  2. learnt based on matrix factorisation techniques
  3. in practical, however, both these models give similar results for many tasks
  4. factors such as *the dataset* on which these models are trained, length of the vectors and so on seem to have a bigger impact than the models themselves

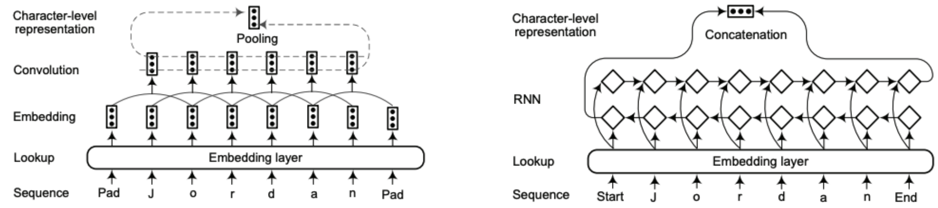
- Facebook fastText
    1. a library created by the Facebook Research Team for efficient learning of word representations and sentence classification
    2. treats each word as composed of character ngrams
    3. so the vector for a word is made of the sum of this character n grams
    4. more computation time, more accurate than word2vec
    5. solve Out-Of-Vocabulary (OOV) issue
    6. the choice of hyper parameters for generating FasText embeddings becomes key
- https://medium.com/analytics-vidhya/word-embeddings-in-nlp-word2vec-glove-fasttext-24d4d4286a73
- https://medium.com/swlh/a-quick-overview-of-the-main-difference-between-word2vec-and-fasttext-b9d3f6e274e9

Out-Of-Vocabulary (OOV)

- words that are unknown by the models
- can be new words or that are derived from spelling errors

character-level representation

- useful for exploiting explicit sub-word-level information such as prefix and suffix
- handles out-of-vocabulary
- two widely-used architectures
  - CNN-based model
    - main difference between CNN and RNN is the ability to process temporal information or data that comes in sequences
    - employ filters within convolutional layers to transform data
    - Whereas, RNNs reuse activation functions from other data points in the sequence to generate the next output in a series
  - RNN-based model
    - CharNER



(a) CNN-based character-level representation.

(b) RNN-based character-level representation.

Fig. 3. CNN-based and RNN-based models for extracting character-level representation for a word.

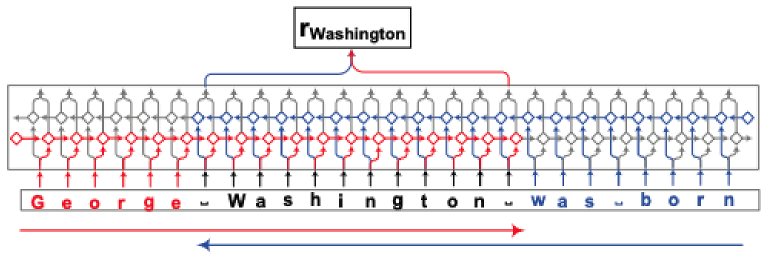


Fig. 4. Extraction of a contextual string embedding for word "Washington" in a sentential context [107]. From the forward language model (shown in red), the model extracts the output hidden state after the last character in the word. From the backward language model (shown in blue), the model extracts the output hidden state before the first character in the word. Both output hidden states are concatenated to form the final embedding of a word.

- a character-level tagger
- considers a sentence as a sequence of characters

- outputs a tag distribution for each character instead of each word

- then word-level tags are obtained from the character-level tags

- taking characters as the primary representation is superior to words as the basic input unit

- bidirectional LSTM

- the embeddings are contextualised by their surrounding text

• same word has different embeddings depending on its contextual use • https://lionbridge.ai/articles/difference-between-cnn-and-rnn/	
context encoder architecture • capture the context dependencies using CNN, RNN or other networks	

1. convolutional neural networks
    a. ID-CNNs achieves 14-20x test-time speedups compared to Bi-LSTM-CRF while retaining comparable accuracy
2. recurrent neural networks
    a. demonstrated remarkable achievements in modelling sequential data
    b. bidirectional RNNs become standard for composing deep context-dependent representations of text
3. recursive neural networks
    a. non-linear adaptive models that are able to learn deep structured information, by traversing a given structure in topological order
    b. named entities are highly related to linguistic constituents
    c. however, typical sequential labelling approaches take little into consideration about phrase structures of sentences
4. neural language model
5. deep transformer

Softmax function

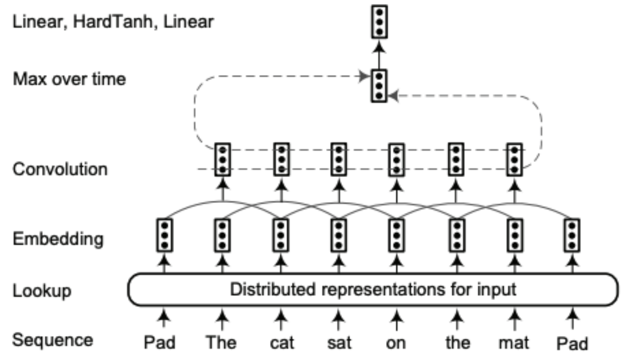


Fig. 5. Sentence approach network based on CNN [17]. The convolution layer extracts features from the whole sentence, treating it as a *sequence* with global structure.

- each word in the input sequence is embedded
- then a convolutional layer is used to produce local features around each word
- the global feature vector is constructed by combining local feature vectors
- two approaches widely used to extract global features
    - max over the position
    - averaging over the position in the sentence
- global features are fed into tag decoder to compute distribution scores for all possible tags for the words

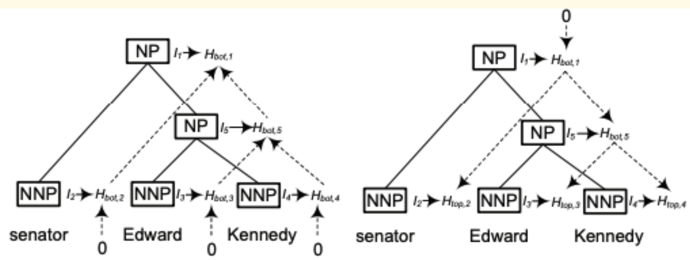


Fig. 8. Bidirectional recursive neural networks for NER [98]. The computations are done recursively in two directions. The bottom-up direction computes the semantic composition of the subtree of each node, and the top-down counterpart propagates to that node the linguistic structures which contain the subtree.

- the bottom-up direction calculates the semantic composition of the subtree of each node
- the top-down counterpart propagates to that node the linguistic structures which contain the subtree
- given hidden vectors for every node, the network calculates a probability distribution of entity types plus a special non-ent

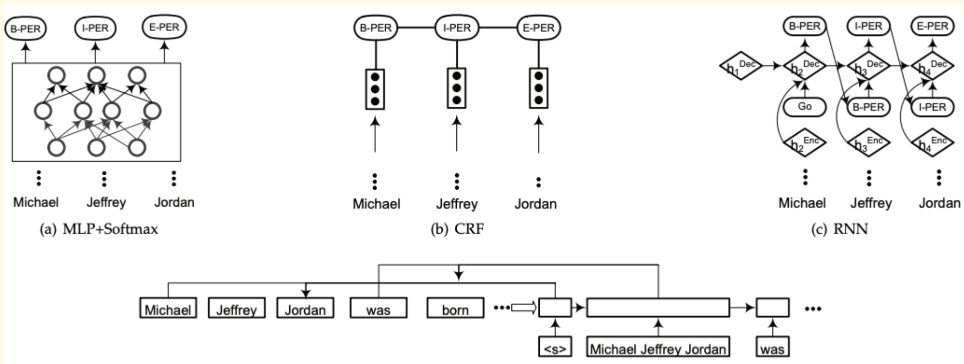
<table>
<tr>
<td></td>
<td>

a generalisation of the logistic function to multiple dimensions
is often used as the last activation function of a neural network to normalise the output of a network to a probability distribution over predicted output classes

</td>
<td></td>
</tr>
<tr>
<td></td>
<td>
tag decoder

takes context-dependent representations as input and
produces a sequence of tags corresponding to the input sequence
4 architectures of tag decoders

multi-layer perceptron + softmax
<ol type="a">
tag for each word is independently predicted based on the context-dependent representations without taking into account its neighbours
gives the probability of the word being in any of the classifications

</td>
<td>


(a) MLP+Softmax (b) CRF (c) RNN (d) Pointer Network
</td>
</tr>
</table>

2. Conditional Random Fields (CRF)
    a. input data is sequential and take previous context into account when making predictions on a data point
    b. the most common choice for tag decoder, and the state-of-the-art performance on CoNLL03 and OntoNotes 5.0
3. recurrent neural networks
    a. faster to train when the number of entity types is large
    b. decoder computes current decoder hidden state in terms of previous step tag, previous step decoder hidden state, and current step encoder hidden state
    c. output tag is decided by using a softmax loss function and is fed to the next step
4. pointer networks
    a. represents variable length dictionaries by using a softmax probability distribution as a 'pointer'

summary

1. pre-trains a bi-directional transformer model in a cloze-style manner, achieves the state-of-the-art performance (93.5%) on CoNLL03
2. BERT and dice loss achieves the state-of-the-art performance (92.07%) on OntoNotes5.0
3. the success of a NER system heavily relies on its input representation
4. integrating or fine-tuning pre-trained language model embeddings is becoming a new paradigm for neural NER

Experiment 1

- article : Training Deep Learning based Named Entity Recognition from Scratch : Disease Extraction Hackathon
  - link : https://appliedmachinelearning.blog/2019/04/01/training-deep-learning-based-named-entity-recognition-from-scratch-disease-extraction-hackathon/
  - code : https://github.com/abhijeet3922/NER_Disease_Extraction_Hackathon/blob/master/Disease_extraction_NER_hackathon_24092019.ipynb

training data	experiment	method	evaluation	output

"Innoplexus Online Hiring Hackathon: Saving lives with AI" which was named entity recognition task
• clinical narratives dataset
  ◦ i.e. : *We compared the inter-day reproducibility of post-occlusive* ***reactive hyperemia*** *(PORH) assessed by single-point laser Doppler flowmetry (LDF) and laser speckle contrast analysis (LSCI).*
• training set : 30,000 documents with labelled entities
• test set : 20,000 document without label
• download : https://www.dropbox.com/s/ef5g11fdq7igi74/hackathon_disease_extraction.zip?dl=0
	• purpose : extract all disease names from a given clinic documents
	• train a custom NER model using Keras(open-source software library provides a Python interface for artificial neural networks)
		• model
		  ◦ embedding : one-hot encoding
		  ◦ dropout
		    ▪ is a regularisation method that approximates training a large number of neural networks with different architectures in parallel
		    ▪ during training, some number of layer outputs are randomly ignored or "dropped out."
		    ▪ https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/
		  ◦ Bi-LSTM : context encoding
		  ◦ time distributed dense
		    ▪ apply the same dense layer (same weights) to the LSTMs outputs for one time step at a time
		    ▪ in this way, the output layer only needs one connection to each LSTM unit
		    ▪ allow the problem to be learned as it was defined, that is one input to one output, keeping the internal process for each time step separate
		    ▪ simplifies the network by requiring far fewer weights such that only one time step is processed at a time
		    ▪ https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/
			77% accuracy on the test data set	tag : inside-outside-beginning (IOB) tagging format

Data

id	doc_id	sent_id	word	tag
1	1	1	Obesity	o
2	1	1	in	o
3	1	1	Low-	o

Experiment 2

- article : Complete Tutorial on Named Entity Recognition (NER) using Python and Keras
  - link : https://www.aitimejournal.com/@akshay.chavan/complete-tutorial-on-named-entity-recognition-ner-using-python-and-keras
  - code : https://github.com/Akshayc1/named-entity-recognition/blob/master/NER%20using%20Bidirectional%20LSTM%20-%20CRF%20.ipynb

training data	experiment	method	evaluation	output

annotated corpus for named entity recognition
• using GMB(Groningen Meaning Bank) corpus
  ◦ GMB is a dataset of multi-sentence texts, together with annotations for parts-of-speech, named entities, lexical categories and other natural language structural phenomena
• tags (BIO scheme)
  ◦ geo = Geographical Entity
  ◦ org = Organisation
  ◦ per = Person
  ◦ gpe = Geopolitical Entity
  ◦ tim = Time indicator
  ◦ art = Artifact
  ◦ eve = Event
  ◦ nat = Natural Phenomenon
• total words count = 1354149
• target data column : "tag"
• download : https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus?select=ner_dataset.csv
	• train the model
	• pick the sentence randomly from test data and predict the label
		• scikit-learn
		• Bi-LSTM
		• CRF
			precision, recall and f1-score metrics	tag

Data

sentence #	word	POS	tag
Sentence: 1	`Thousands`	`NNS`	o
	of	IN	o
	demonstrators	NNS	o

MasakhaNER: Named Entity Recognition for African Languages

- Goal : analyse ten African languages and conduct an extensive empirical evaluation of state--of--the--art methods across both supervised and transfer learning settings
- Link : https://research.google/pubs/pub50293/ (03/2021)
- Code : https://github.com/masakhane-io/masakhane-ner
- Credible source: Google Research

Model

type	model	description	equation /pseudo code

baseline	CNN--BiLSTM--CRF	• For each input sequence, we first compute the vector representation for each word by concatenating character--level encodings from a CNN and vector embeddings for each word.
		• Conditional Random Field(CRF)
		  ◦ a class of statistical modelling method often applied in pattern recognition and machine learning
		  ◦ used for structured prediction
		  ◦ take context into account
		  ◦ whereas a classifier predicts a label for a single sample without considering 'neighbouring' samples	

	multilingual BERT (mBERT)		
	XLM--RoBERTa (XLM-R)		
	MeanE--BiLSTM		
improving the baseline	Gazetteers for NER		
	transfer learning from another domain		
	aggregating NER datasets by regions		

Experiment

training data	experiment	method	evaluation	output
African NER datasets • 10 languages from different data source				

Repository

Named Entity Recognition as Dependency Parsing

- Goal : use ideas from graph-based dependency parsing to provide our model a global view on the input via a biaffine model
- Link : https://research.google/pubs/pub49152/(06/2020)
- Code : https://github.com/juntaoy/biaffine-ner
- Credible source: Google Research

A Joint Named-Entity Recogniser for Heterogeneous Tag-sets Using a Tag Hierarchy

- Goal : shows the benefit of the tag-hierarchy model, especially when facing non-trivial consolidation of tag-sets
- Link : https://research.google/pubs/pub48896/ (07/2019)
- Code : -
- Credible source: Google Research

Example-Based Named Entity Recognition

- Goal : present a novel approach to named entity recognition in the presence of scarce data that we call example-based NER
- Link :https://arxiv.org/abs/2008.10570 (08/2020)
- Code : -
- Credible -

Domain-Transferable Method for Named Entity Recognition Task

- Goal : describes a method to learn a domain-specific NER model for an arbitrary set of named entities when domain-specific supervision is not available
- Link : https://arxiv.org/abs/2011.12170 (11/2020)
- Code : https://github.com/vmkhlv/histqa-domain-ner
- Credible -

FLERT: Document-Level Features for Named Entity Recognition

- Goal : perform a comparative evaluation of document-level features in the two standard NER architectures commonly considered in the literature, namely ”fine-tuning” and ”feature-based LSTM-CRF”
- Link : https://arxiv.org/pdf/2011.06993v2.pdf (05/2021)
- Code : https://github.com/flairNLP/flair
- Credible source: Humboldt University of Berlin