In [5]:

```
%%html
<style>
table {display: block;}
td {
  font-size: 18px
}
.rendered_html { font-size: 28px; }
*{ line-height: 200%; }
</style>

```

Natural Language Processing and the Web WS 2022/23- Practice

Class - Tutorial 6

In this practice class, we will further discuss the supervised machine learning approach and feature generation/selection strategies. We will also discuss basics of PyTorch, a deep learning framework in Python and how to build NLP applications using such models. We will also briefly discuss on how to serve ML models in web application using the Flask framewrok.

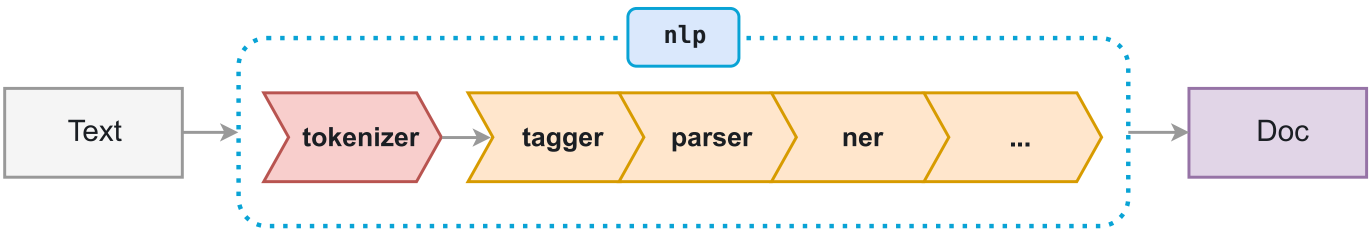
At the end of this notebook, there are 4 machine learning assignment descriptions. Please form a group (2 - 5 students) and choose one of the assignments. The assignment is due in two weeks.

Contents

[How pipelines work (https://spacy.io/usage/processing-pipelines#pipelines)](https://spacy.io/usage/processing-pipelines#pipelines)

- Re-use existing libraries
- Custom component can be added when initializing a `Language` class

When you load a model, spaCy first consults the model's meta.json. The meta typically includes the model details, the ID of a language class, and an optional list of pipeline components.



Adding custom pipeline

component - entittyMatcher

In [46]:

```
import spacy
from spacy.pipeline import EntityRuler
nlp = spacy.load('en_core_web_sm')

terms =  ["cat", "dog", "artic foxes"]
ruler = nlp.add_pipe("entity_ruler")
for t in terms:
    ruler.add_patterns([{"label": "ANIMAL", "pattern": t}])

doc = nlp("There is no cat in the house and no artic foxes in the basement: Fox
print([(ent.start, ent.end, ent.text, ent.lemma_, ent.label_) for ent in doc.ent
```

```
[(3, 4, 'cat', 'cat', 'ANIMAL'), (9, 11, 'artic foxes', 'artic fox',
'ANIMAL'), (15, 17, 'Fox News', 'Fox News', 'ORG')]
```

Training new ner Model - toy examples for product name recognition

In [3]:

```python
import spacy
from tqdm import tqdm # loading bar
from spacy.training.example import Example
import random
TRAIN_DATA = [('what is the price of polo?', {'entities': [(21, 25, 'PrdName')]}
              ('what is the price of ball?', {'entities': [(21, 25, 'PrdName')]}
              ('what is the price of jegging?', {'entities': [(21, 28, 'PrdName'
              ('what is the price of t-shirt?', {'entities': [(21, 28, 'PrdName'
              ('what is the price of jeans?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of bat?', {'entities': [(21, 24, 'PrdName')]})
              ('what is the price of shirt?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of bag?', {'entities': [(21, 24, 'PrdName')]})
              ('what is the price of cup?', {'entities': [(21, 24, 'PrdName')]})
              ('what is the price of jug?', {'entities': [(21, 24, 'PrdName')]})
              ('what is the price of plate?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of glass?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of moniter?', {'entities': [(21, 28, 'PrdName'
              ('what is the price of desktop?', {'entities': [(21, 28, 'PrdName'
              ('what is the price of bottle?', {'entities': [(21, 27, 'PrdName')
              ('what is the price of mouse?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of keyboad?', {'entities': [(21, 28, 'PrdName'
              ('what is the price of chair?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of table?', {'entities': [(21, 26, 'PrdName')]
              ('what is the price of watch?', {'entities': [(21, 26, 'PrdName')]

def train_spacy(data,iterations):
    TRAIN_DATA = data
    nlp = spacy.blank('en')  # create blank Language class
    # create the built-in pipeline components and add them to the pipeline
    # nlp.create_pipe works for built-ins that are registered with spaCy
    if 'ner' not in nlp.pipe_names:
        ner = nlp.create_pipe('ner')
        nlp.add_pipe('ner', last=True)
    # add labels
    for _, annotations in TRAIN_DATA:
```

```python
         for ent in annotations.get('entities'):
            ner.add_label(ent[2])

    # get names of other pipes to disable them during training
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
    with nlp.disable_pipes(*other_pipes):  # only train NER
        optimizer = nlp.begin_training()
        for itn in range(iterations):
            print("Statring iteration " + str(itn))
            random.shuffle(TRAIN_DATA)
            losses = {}
            for text, annotations in  tqdm(TRAIN_DATA):
                doc = nlp.make_doc(text)
                example = Example.from_dict(doc, annotations)
                nlp.update([example],
                    drop=0.2,  # dropout - make it harder to memorise data
                    sgd=optimizer,  # callable to update weights
                    losses=losses)
            print("losses", losses)
    return nlp


prdnlp = train_spacy(TRAIN_DATA, 20)

# Save our trained Model
modelfile = "spacy_prdName"
prdnlp.to_disk(modelfile)

#Test your text
test_text = "what is the price of chair?"
doc = prdnlp(test_text)
print("\n=========Test resultt======\n")
for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

```
 30%|███       | 6/20 [00:00<00:00, 56.45it/s]
Statring iteration 0
```

```
100%|██████████| 20/20 [00:00<00:00, 61.33it/s]
 35%|███▌      | 7/20 [00:00<00:00, 65.55it/s]

losses {'ner': 52.84350101363053}
Statring iteration 1

100%|██████████| 20/20 [00:00<00:00, 65.97it/s]
 35%|███▌      | 7/20 [00:00<00:00, 65.00it/s]

losses {'ner': 7.238427457959996}
Statring iteration 2

100%|██████████| 20/20 [00:00<00:00, 65.49it/s]
 35%|███▌      | 7/20 [00:00<00:00, 66.26it/s]

losses {'ner': 3.9251932809686414}
Statring iteration 3
```

In [4]:

```python
for tok in doc:
    print(tok.text, tok.lemma_)
```

```
what
is
the
price
of
chair
?
```

In [5]:

```python
!which python
```

```
/Users/abhikjana/anaconda3/bin/python
```

In [6]:

```
# load the German model - you can use this if you do the GermaNER assignment (se
# Restart the kernel to use the model, if this is the first time you download th
!python -m spacy download de_core_news_sm
```

```
WARNING: Ignoring invalid distribution -cipy (/Users/abhikjana/anacon
da3/lib/python3.7/site-packages)
WARNING: Ignoring invalid distribution -cipy (/Users/abhikjana/anacon
da3/lib/python3.7/site-packages)
Collecting de-core-news-sm==3.1.0
  Downloading https://github.com/explosion/spacy-models/releases/down
load/de_core_news_sm-3.1.0/de_core_news_sm-3.1.0-py3-none-any.whl (ht
tps://github.com/explosion/spacy-models/releases/download/de_core_new
s_sm-3.1.0/de_core_news_sm-3.1.0-py3-none-any.whl) (18.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.8/18.8 MB 11.0 MB/s
eta 0:00:0000:0100:01
Requirement already satisfied: spacy<3.2.0,>=3.1.0 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from de-core-news-sm==3.1.0)
(3.1.4)
Requirement already satisfied: thinc<8.1.0,>=8.0.12 in /Users/abhikja
na/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->d
e-core-news-sm==3.1.0) (8.0.13)
Requirement already satisfied: numpy>=1.15.0 in /Users/abhikjana/anac
onda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de-core-
news-sm==3.1.0) (1.19.1)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4 i
n /Users/abhikjana/anaconda3/lib/python3.7/site-packages (from spacy<
3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (1.8.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.8 in /Users/a
bhikjana/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.
1.0->de-core-news-sm==3.1.0) (3.0.8)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /Users/ab
hikjana/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.
1.0->de-core-news-sm==3.1.0) (1.0.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de
-core-news-sm==3.1.0) (4.48.0)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de
-core-news-sm==3.1.0) (0.4.1)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /Users/abhikj
ana/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->
de-core-news-sm==3.1.0) (3.0.2)
Requirement already satisfied: pathy>=0.3.5 in /Users/abhikjana/anaco
nda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de-core-n
ews-sm==3.1.0) (0.6.1)
Requirement already satisfied: packaging>=20.0 in /Users/abhikjana/an
aconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de-cor
e-news-sm==3.1.0) (21.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /Users/abhi
kjana/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0
```

```
->de-core-news-sm==3.1.0) (2.24.0)
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de
-core-news-sm==3.1.0) (2.4.2)
Requirement already satisfied: typing-extensions<4.0.0.0,>=3.7.4 in /
Users/abhikjana/anaconda3/lib/python3.7/site-packages (from spacy<3.
2.0,>=3.1.0->de-core-news-sm==3.1.0) (3.10.0.2)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de
-core-news-sm==3.1.0) (0.4.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de
-core-news-sm==3.1.0) (2.0.3)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /Users/abhikja
na/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->d
e-core-news-sm==3.1.0) (0.8.2)
Requirement already satisfied: setuptools in /Users/abhikjana/anacond
a3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de-core-new
s-sm==3.1.0) (41.4.0)
Requirement already satisfied: jinja2 in /Users/abhikjana/anaconda3/l
ib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->de-core-news-sm
==3.1.0) (2.10.3)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /Users/abhi
kjana/anaconda3/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0
->de-core-news-sm==3.1.0) (2.0.6)
Requirement already satisfied: zipp>=0.5 in /Users/abhikjana/anaconda
3/lib/python3.7/site-packages (from catalogue<2.1.0,>=2.0.6->spacy<3.
2.0,>=3.1.0->de-core-news-sm==3.1.0) (0.6.0)
Requirement already satisfied: pyparsing<3,>=2.0.2 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from packaging>=20.0->spacy<
3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (2.4.2)
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /Users/abh
ikjana/anaconda3/lib/python3.7/site-packages (from pathy>=0.3.5->spac
y<3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (5.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0
->spacy<3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (2019.9.11)
Requirement already satisfied: idna<3,>=2.5 in /Users/abhikjana/anaco
nda3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0->spacy
<3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (2.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /Users/abhikjana/
anaconda3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0->
spacy<3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.
1 in /Users/abhikjana/anaconda3/lib/python3.7/site-packages (from req
uests<3.0.0,>=2.13.0->spacy<3.2.0,>=3.1.0->de-core-news-sm==3.1.0)
 (1.24.2)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /Users/abhikjan
a/anaconda3/lib/python3.7/site-packages (from typer<0.5.0,>=0.3.0->sp
acy<3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /Users/abhikjana/a
naconda3/lib/python3.7/site-packages (from jinja2->spacy<3.2.0,>=3.1.
0->de-core-news-sm==3.1.0) (1.1.1)
Requirement already satisfied: more-itertools in /Users/abhikjana/ana
conda3/lib/python3.7/site-packages (from zipp>=0.5->catalogue<2.1.0,>
=2.0.6->spacy<3.2.0,>=3.1.0->de-core-news-sm==3.1.0) (7.2.0)
WARNING: Ignoring invalid distribution -cipy (/Users/abhikjana/anacon
da3/lib/python3.7/site-packages)
WARNING: Ignoring invalid distribution -cipy (/Users/abhikjana/anacon
```

```
da3/lib/python3.7/site-packages)
WARNING: Ignoring invalid distribution -cipy (/Users/abhikjana/anacon
da3/lib/python3.7/site-packages)
WARNING: Ignoring invalid distribution -cipy (/Users/abhikjana/anacon
da3/lib/python3.7/site-packages)

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: pip3.7 install --upgrade pip
✔ Download and installation successful
You can now load the package via spacy.load('de_core_news_sm')
```

In [7]:

```python
import spacy
from spacy.lang.de.examples import sentences

nlp = spacy.load('de_core_news_sm')

doc = nlp(sentences[0])
print(doc.text)
for ent in doc.ents:
    print(ent, ent.label_)
```

```
Die ganze Stadt ist ein Startup: Shenzhen ist das Silicon Valley für H
ardware-Firmen
Shenzhen LOC
Silicon Valley LOC
Hardware-Firmen LOC
```

Sequence labeling using Conditional Random Field

(CRF) - pycrfsuite (https://albertauyeung.githul sequence-labelling-with-crf.html/)

CRF is a probabilistic graphical model that can be used to model sequential data. The feature fucntion extracts features for each word in a sentence. During model training, CRF will try to determine the weights of different feature functions that will maximise the likelihood of the labels in the training data.

One of the commonly used CRF library is CRFSuite implemented by Naoaki Okazaki in C/C++. The python wrapper for this model is pycrfsuite.

In [8]:

```python
# Define features usefull for POS tagging
# Features: Word itself, 2 and 3 letter suffixes, previous and next word
# From here: https://nlpforhackers.io/training-pos-tagger/
def features(sentence, index):
    """ sentence: [w1, w2, ...], index: the index of the word """
    features =  [
        'word='+ sentence[index],
        'is_capitalized='+ str(sentence[index][0].upper() == sentence[index][0])
        'is_all_caps='+ str(sentence[index].upper() == sentence[index]),
        'is_all_lower='+ str(sentence[index].lower() == sentence[index]),
        'prefix-1='+ sentence[index][0],
        'prefix-2='+ sentence[index][:2],
        'prefix-3='+ sentence[index][:3],
        'suffix-1='+ sentence[index][-1],
        'suffix-2='+ sentence[index][-2:],
        'suffix-3='+ sentence[index][-3:],
        'prev_word='+ str( '' if index == 0 else sentence[index - 1]),
        'next_word='+ str('' if index == len(sentence) - 1 else sentence[index +
        'has_hyphen='+ str('-' in sentence[index]),
        'is_numeric='+ str(sentence[index].isdigit()),
        'capitals_inside='+ str(sentence[index][1:].lower() != sentence[index][1
    ]
    if index == 0:
        features.append("BOS")
    if index == len(sentence) - 1:
        features.append("EOS")
    return features


import pprint
# Show how the extracted POS features looks like
pprint.pprint(features(['This', 'is', 'a', 'sentence'], 0))
```

```
['word=This',
 'is_capitalized=True',
 'is_all_caps=False',
 'is_all_lower=False',
```

```
 'prefix-1=T',
 'prefix-2=Th',
 'prefix-3=Thi',
 'suffix-1=s',
 'suffix-2=is',
 'suffix-3=his',
 'prev_word=',
 'next_word=is',
 'has_hyphen=False',
 'is_numeric=False',
 'capitals_inside=False',
 'BOS']
```

In [9]:

```python
# Create features for words in each sentence.
def transform_file_to_dataset(filename):
    #arrays features of all tokens in a sentence
    all_sents, all_poses = [], []
    with open(filename) as file:
        tokens = []
        tokenposes = []
        sents = []
        sentposes =[]
        for line in file:
            if not line.strip():
                for index in range(len(tokens)):
                    sents.append(features(tokens, index))
                    sentposes.append(tokenposes[index])
                all_sents.append(sents)
                all_poses.append(sentposes)
                sents,sentposes,tokens,tokenposes = [],[],[],[]
            else:
                lines = line.split("\t")
                tokens.append(lines[0])
                tokenposes.append(lines[1].strip())
    return all_sents, all_poses
```

In [10]:

```
from sklearn.model_selection import train_test_split
pos_features, tags = transform_file_to_dataset("data/pos_data/wsj_pos.dev")
pos_features_test, tags_test = transform_file_to_dataset("data/pos_data/wsj_pos.
```

In [11]:

```
# features for the first sentence
pos_features[0]
```

Out[11]:

```
[['word=The',
  'is_capitalized=True',
  'is_all_caps=False',
  'is_all_lower=False',
  'prefix-1=T',
  'prefix-2=Th',
  'prefix-3=The',
  'suffix-1=e',
  'suffix-2=he',
  'suffix-3=The',
  'prev_word=',
  'next_word=Arizona',
  'has_hyphen=False',
  'is_numeric=False',
  'capitals_inside=False',
  'BOS'],
 ['word=Arizona',
```

In [12]:

```
# Install The pytthon CRFsuit model
!pip install python-crfsuite
```

```
Requirement already satisfied: python-crfsuite in /Library/Frameworks/
Python.framework/Versions/3.8/lib/python3.8/site-packages (0.9.8)

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: pip install --upgrade pip
```

In [13]:

```python
import pycrfsuite
trainer = pycrfsuite.Trainer(verbose=True)
# Submit training data to the trainer
for xseq, yseq in zip(pos_features, tags):
    trainer.append(xseq, yseq)

# Set the parameters of the model
trainer.set_params({
    # coefficient for L1 penalty
    'c1': 0.1,

    # coefficient for L2 penalty
    'c2': 0.01,

    # maximum number of iterations
    'max_iterations': 200,

    # whether to include transitions that
    # are possible, but not observed
    'feature.possible_transitions': True
})

# Provide a file name as a parameter to the train function, such that
# the model will be saved to the file when training is finished
trainer.train('crf.model')
```

```
Feature generation
type: CRF1d
feature.minfreq: 0.000000
feature.possible_states: 0
feature.possible_transitions: 1
0....1....2....3....4....5....6....7....8....9....10
Number of features: 106844
Seconds required: 0.842

L-BFGS optimization
c1: 0.100000
c2: 0.010000
num_memories: 6
max_iterations: 200
epsilon: 0.000010
stop: 10
```

```
delta: 0.000010
linesearch: MoreThuente
linesearch.max_iterations: 20
```

In [14]:

```python
# Loading the tagger and predict
tagger = pycrfsuite.Tagger()
tagger.open('crf.model')
y_pred = [tagger.tag(pos_feature_test) for pos_feature_test in pos_features_test
```

In [15]:

```python
# Let's take a look at a random sample in the testing set sentence
i = 23
# rhe feature set for instance "i"
print (pos_features_test[i])
print("\n======\n")
# The word, the gold label and the predicted value
for x,  g, y in zip([x[0].split("=")[1] for x in pos_features_test[i]],  tags_te
    print("%s (%s) (%s)" % (x, y, g))
```

```
[['word=PaineWebber', 'is_capitalized=True', 'is_all_caps=False', 'is
_all_lower=False', 'prefix-1=P', 'prefix-2=Pa', 'prefix-3=Pai', 'suff
ix-1=r', 'suffix-2=er', 'suffix-3=ber', 'prev_word=', 'next_word=In
c.', 'has_hyphen=False', 'is_numeric=False', 'capitals_inside=True',
'BOS'], ['word=Inc.', 'is_capitalized=True', 'is_all_caps=False', 'is
_all_lower=False', 'prefix-1=I', 'prefix-2=In', 'prefix-3=Inc', 'suff
ix-1=.', 'suffix-2=c.', 'suffix-3=nc.', 'prev_word=PaineWebber', 'nex
t_word=filmed', 'has_hyphen=False', 'is_numeric=False', 'capitals_ins
ide=False'], ['word=filmed', 'is_capitalized=False', 'is_all_caps=Fal
se', 'is_all_lower=True', 'prefix-1=f', 'prefix-2=fi', 'prefix-3=fi
l', 'suffix-1=d', 'suffix-2=ed', 'suffix-3=med', 'prev_word=Inc.', 'n
ext_word=a', 'has_hyphen=False', 'is_numeric=False', 'capitals_inside
=False'], ['word=a', 'is_capitalized=False', 'is_all_caps=False', 'is
_all_lower=True', 'prefix-1=a', 'prefix-2=a', 'prefix-3=a', 'suffix-1
=a', 'suffix-2=a', 'suffix-3=a', 'prev_word=filmed', 'next_word=new',
'has_hyphen=False', 'is_numeric=False', 'capitals_inside=False'], ['w
ord=new', 'is_capitalized=False', 'is_all_caps=False', 'is_all_lower=
True', 'prefix-1=n', 'prefix-2=ne', 'prefix-3=new', 'suffix-1=w', 'su
ffix-2=ew', 'suffix-3=new', 'prev_word=a', 'next_word=television', 'h
```

In [16]:

```
# Show gold and prediction side by side for sent[100],
# See that the third word is wrongly taged as POS in stead of NNP
for sent_g, sent_p in zip(tags_test[100], y_pred[100]):
    print(sent_g, sent_p)

```

```
DT DT
NNP NNP
NNP POS
NN NN
IN IN
DT DT
JJ JJ
NN NN
VBD VBD
NNP NNP
NNP NNP
, ,
WP WP
VBZ VBZ
DT DT
JJ JJ
NN NN
IN IN
DT DT
JJ JJ
NN NN
. .
```

In [17]:

```
#Reporting the performance using the sklearn precision_recall_fscore support
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.metrics import precision_recall_fscore_support
multi = MultiLabelBinarizer()

test_new = multi.fit(tags_test).transform(tags_test)
pred_new = multi.transform(y_pred)
print("samples",precision_recall_fscore_support(pred_new, test_new, average="sam
print("macro", precision_recall_fscore_support(pred_new, test_new, average="macr
```

```
samples (0.9720857648345673, 0.9710564612404172, 0.9708756036159525, N
one)
macro (0.8630728070312608, 0.8778053970140426, 0.8623640913846606, Non
e)
```

```
/Users/abhikjana/anaconda3/lib/python3.7/site-packages/sklearn/metric
s/classification.py:1439: UndefinedMetricWarning: Recall and F-score a
re ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

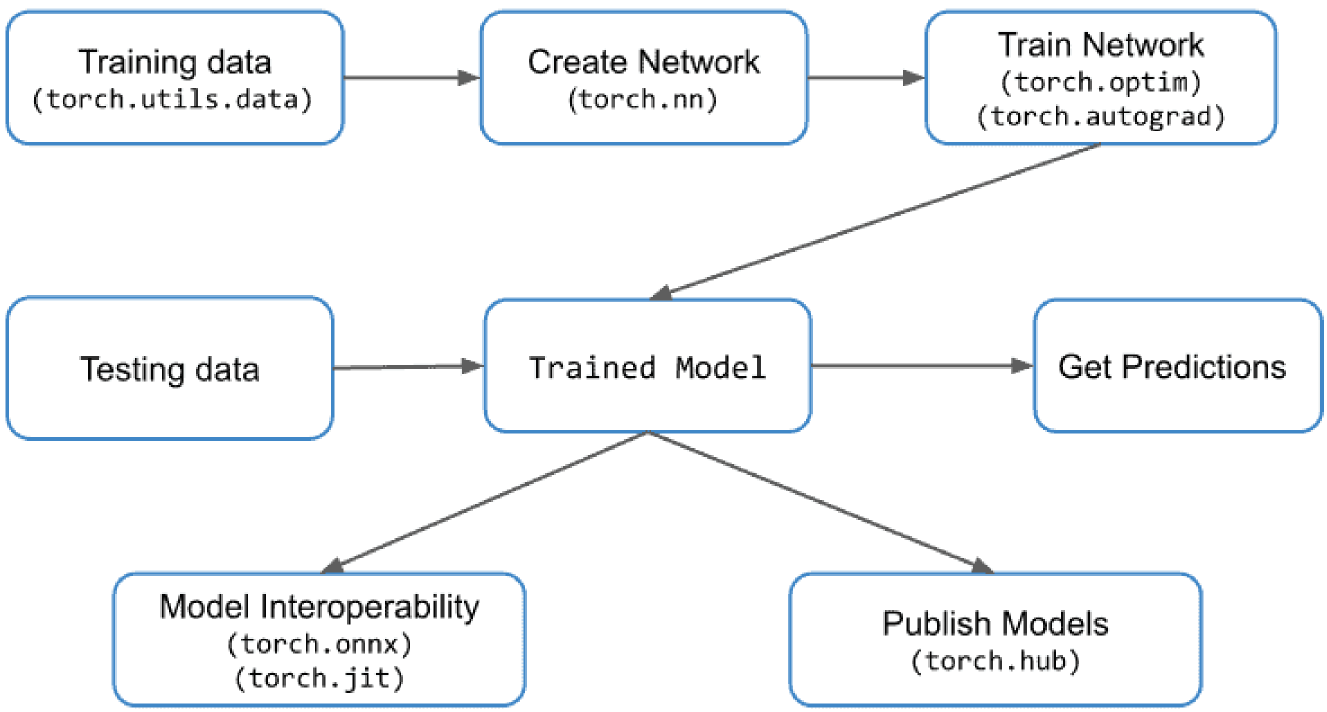
Introduction to PyTorch

PyTorch is a python based library that facilitate building deep learning models. It incorporates an advanced scientific computing capability.

PyTorch Tensor

- Similar to Numpy Array but with much more capability and fast computations, support of GPU acceleration, distributed computing, and automatic gradient calculation
- Provides acceleration for various mathematical operations

A typical workflow in PyTorch



Data loading in PyTorch

- Dataset: Built on top of Tensor data type and used for custom datasets
- DataLoader: Is used to load larger dataset

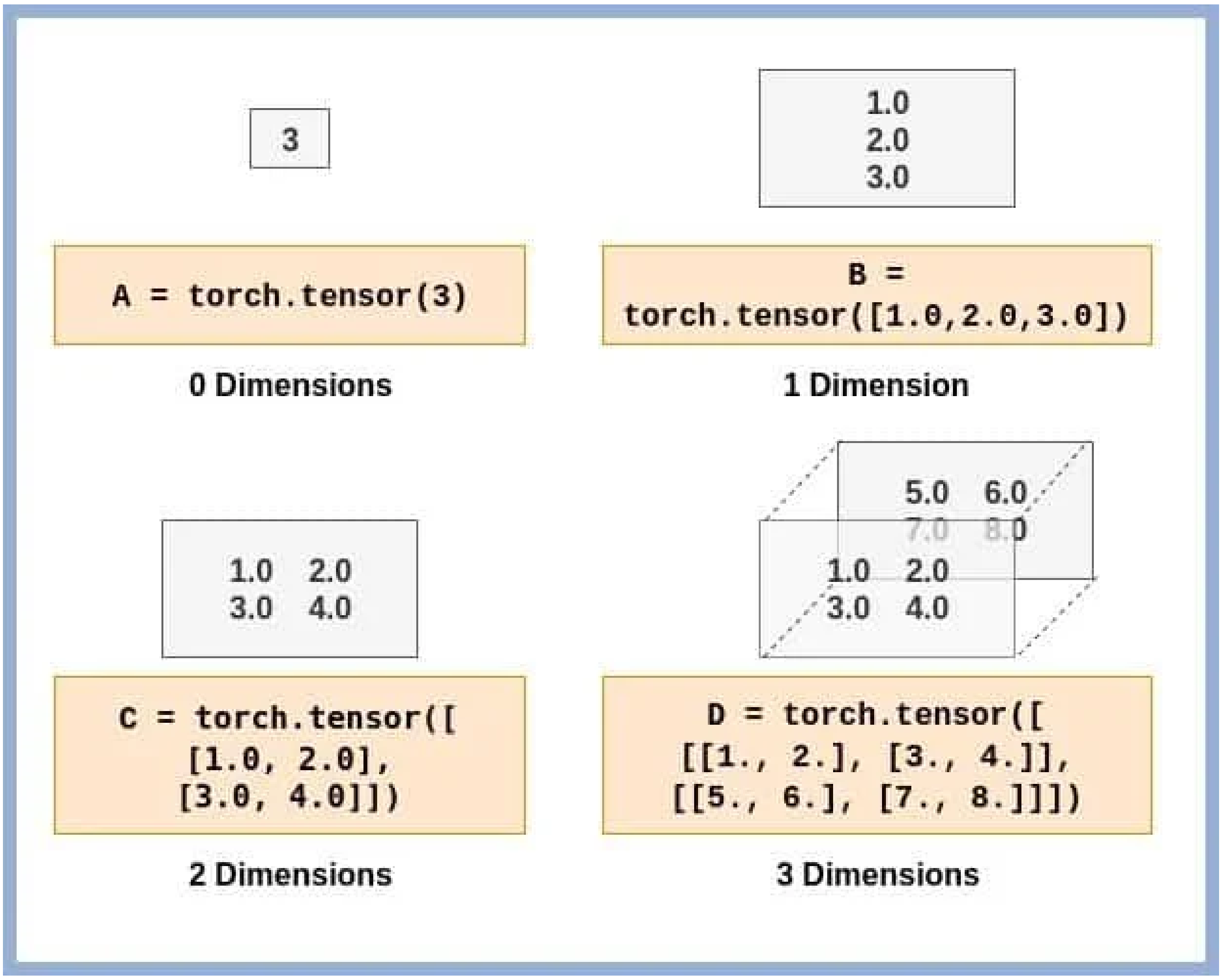
Creating Network

- The `torch.nn` module is used to create Neural networks
- Provide neural netwrok (nn) layers such as fully connected layers, convolutional layers, activation and loss functions
- Once network architecture is created and data are ready to be fed, the `torch.optim` module is used to update weights and biases.
- Automatic differentiation is provided by the `torch.autograd` module

Tensors

Tensors are similar to Numpy, that are used to create

n-dimensional arrays --> Matrices



In [18]:

```
# Install pytorch
!/opt/anaconda3/bin/pip install torch torchvision
```

```
/bin/sh: /opt/anaconda3/bin/pip: No such file or directory
```

Running PyTorch using Google Colab

You can also use [Google Colab (https://colab.research.google.com/)](https://colab.research.google.com/) to build deep learning models, they are free, fast and enough for some experiments that you can not run on your local machine. Read about the GPU and TPU supports [here (https://www.bmc.com/blogs/google-cloud-tpu/)](https://www.bmc.com/blogs/google-cloud-tpu/)

In [19]:

```python
import torch
# one dim tensor
print(torch.ones(10))
print(torch.zeros(5))
a = torch.tensor([1,2,3,4])
print(a)
print(type(a))
```

```
tensor([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
tensor([0., 0., 0., 0., 0.])
tensor([1, 2, 3, 4])
<class 'torch.Tensor'>
```

In [20]:

```
#two dim tensors
print(torch.zeros(2,3))
b = torch.tensor([[1,2,3],
                  [3,4,5],
                  [6,7,8],
                  [9,10,11]])
print(b)
print(b.shape)
# accessing the element in tensor
print(b[2]) # get elements at row 2 (i.e 3)
print(b[:2]) # get rows 0 and 1
print(b[:,2]) # get elements at column 2 (i.e 3 -- the last column)
print(b[2]+2) # add 2 two each element
```

```
tensor([[0., 0., 0.],
        [0., 0., 0.]])
tensor([[ 1,  2,  3],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 9, 10, 11]])
torch.Size([4, 3])
tensor([6, 7, 8])
tensor([[1, 2, 3],
        [3, 4, 5]])
tensor([ 3,  5,  8, 11])
tensor([ 8,  9, 10])
```

In [21]:

```
# specify the data type
a = torch.tensor([1,2,3], dtype=torch.int32) # 32-bit Integer
print(a)
b = torch.tensor([2.,4., 6.], dtype=torch.float64) # 64 bit floating point
print(b)
```

```
tensor([1, 2, 3], dtype=torch.int32)
tensor([2., 4., 6.], dtype=torch.float64)
```

In [22]:

```python
# Tensor to Numpy and vice versa
aa = a.numpy()
print(aa)
print(type(aa))
aaa = torch.from_numpy(aa)
aaa
```

```
[1 2 3]
<class 'numpy.ndarray'>
```

Out[22]:

```
tensor([1, 2, 3], dtype=torch.int32)
```

In [23]:

```
#Arthimetic operations
# Create tensor
tensor1 = torch.tensor([[1,2,3],[4,5,6]])
tensor2 = torch.tensor([[-1,2,-3],[4,-5,6]])

# Addition
print("Addition:",tensor1+tensor2)
# We can also use
print("Addition2:",torch.add(tensor1,tensor2))

# Subtraction
print("Subtraction:",tensor1-tensor2)
# We can also use
print("Subtraction2:",torch.sub(tensor1,tensor2))

# Multiplication
# Tensor with Scalar
print("Mult:",tensor1 * 2)

# Tensor with another tensor
# Elementwise Multiplication
print("Mult two tensors:",tensor1 * tensor2)

# Matrix multiplication
tensor3 = torch.tensor([[1,2],[3,4],[5,6]])
print("Matrix mult:",torch.mm(tensor1,tensor3))

# Division
# Tensor with scalar
print("Division:",tensor1//2)

# Tensor with another tensor
# Elementwise division
print("Elementwise div:",tensor1//tensor2)

```

```
Addition: tensor([[ 0,  4,  0],
        [ 8,  0, 12]])
Addition2: tensor([[ 0,  4,  0],
        [ 8,  0, 12]])
Subtraction: tensor([[ 2,  0,  6],
        [ 0, 10,  0]])
Subtraction2: tensor([[ 2,  0,  6],
        [ 0, 10,  0]])
Mult: tensor([[ 2,  4,  6],
        [ 8, 10, 12]])
Mult two tensors: tensor([[ -1,   4,  -9],
        [ 16, -25,  36]])
Matrix mult: tensor([[22, 28],
        [49, 64]])
Division: tensor([[0, 1, 1],
        [2, 2, 3]])
Elementwise div: tensor([[-1,  1, -1],
        [ 1, -1,  1]])
```

```
/Users/abhikjana/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:30: UserWarning: __floordiv__ is deprecated, and its behavior w
ill change in a future version of pytorch. It currently rounds toward
0 (like the 'trunc' function NOT 'floor'). This results in incorrect r
ounding for negative values. To keep the current behavior, use torch.d
iv(a, b, rounding_mode='trunc'), or for actual floor division, use tor
ch.div(a, b, rounding_mode='floor').
/Users/abhikjana/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:34: UserWarning: __floordiv__ is deprecated, and its behavior w
ill change in a future version of pytorch. It currently rounds toward
0 (like the 'trunc' function NOT 'floor'). This results in incorrect r
ounding for negative values. To keep the current behavior, use torch.d
iv(a, b, rounding_mode='trunc'), or for actual floor division, use tor
ch.div(a, b, rounding_mode='floor').
```

In [24]:

```
#GPU and CPU tensor
# Create a tensor for CPU
# This will occupy CPU RAM
tensor_cpu = torch.tensor([[1.0, 2.0], [3.0, 4.0], [5.0, 6.0]], device='cpu')



# Create a tensor for GPU
# This will occupy GPU RAM
# check if GPU is avaliable
print(torch.cuda.is_available())
if torch.cuda.is_available():
    tensor_gpu = torch.tensor([[1.0, 2.0], [3.0, 4.0], [5.0, 6.0]], device='cuda
```

```
False
```

In [25]:

```
# Reshaping tensors
a = torch.randn(2,4)
print(a)
print(a.shape)
b = a.reshape(1,8)
print(b)
print(b.shape)
```

```
tensor([[ 1.4158, -1.4568,  0.1217, -0.2896],
        [ 1.6169, -0.9942, -0.1312,  0.2329]])
torch.Size([2, 4])
tensor([[ 1.4158, -1.4568,  0.1217, -0.2896,  1.6169, -0.9942, -0.131
2,  0.2329]])
torch.Size([1, 8])
```

In [26]:

```
# Autograd module - Autograd is PyTorch's automatic differentiation package
a = torch.ones((2,2), requires_grad=True)
print("A =", a)
b = a + 5
print("B =", b)
c = b.mean()
print("C =", c)
# back propagating
c.backward()
# computing gradients
print("Gradients-->",a.grad)
```

```
A = tensor([[1., 1.],
        [1., 1.]], requires_grad=True)
B = tensor([[6., 6.],
        [6., 6.]], grad_fn=<AddBackward0>)
C = tensor(6., grad_fn=<MeanBackward0>)
Gradients--> tensor([[0.2500, 0.2500],
        [0.2500, 0.2500]])
```

$$b = a + 5$$

c = mean(b) = Σ(a+5) / 4

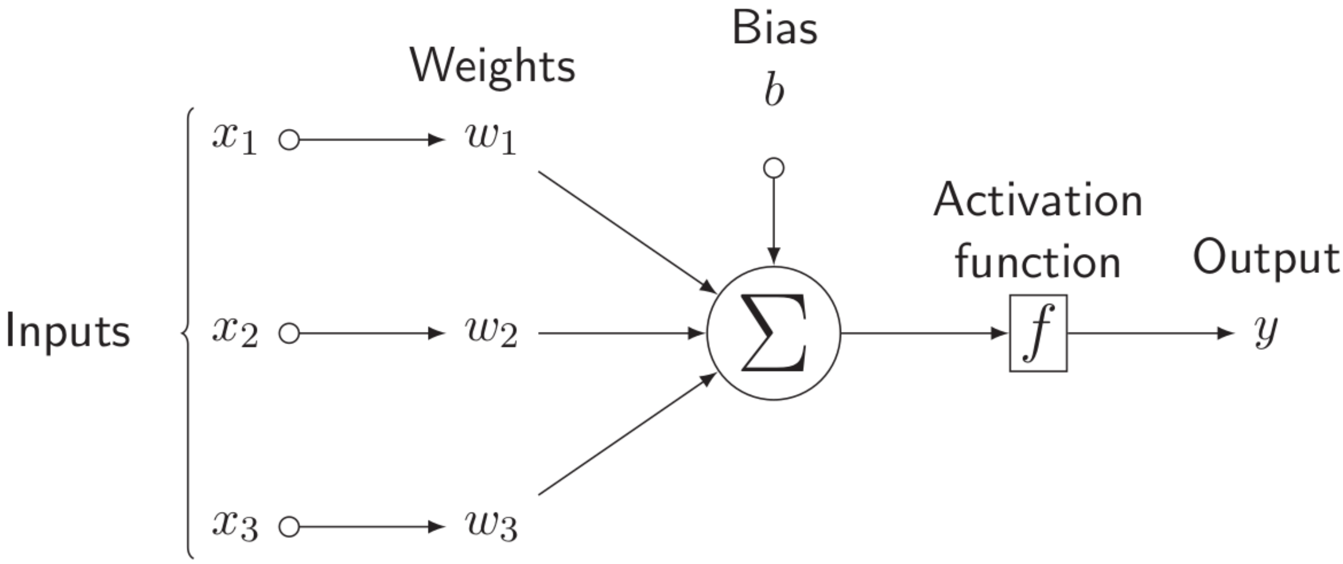
derviation(a) w.r.t c = 1/4

Perceptron: The simplest neural network

Like a biological neuron, there is `input` and `output` where the `signals` flow from the inputs to the the outputs.

$$y = f(wx + b)$$

`x` is the input, `y` is the output, `wx +b` is a linear function while `f(wx+b)` is a non-linear function called the `activation function.`



Activation functions

These are nonlinear functions used to capture complex relationship in data.

Sigmoid

It takes any real value and squashes it into the range between 0 and 1

$$f(x) = \frac{1}{1+e^{-x}}$$

It is mostly used at the output. See the following code for an example.

In [27]:

```python
import torch
import matplotlib.pyplot as plt
import torch.nn as nn
# numbers from -5 to 5 with interval of 0.1 each
x = torch.arange(-6., 6., 0.1)
print(x)
y = torch.sigmoid(x)
print(y)
plt.plot(x.numpy(), y.detach().numpy())
plt.show()
```

```
tensor([-6.0000e+00, -5.9000e+00, -5.8000e+00, -5.7000e+00, -5.6000e+0
0,
        -5.5000e+00, -5.4000e+00, -5.3000e+00, -5.2000e+00, -5.1000e+0
0,
        -5.0000e+00, -4.9000e+00, -4.8000e+00, -4.7000e+00, -4.6000e+0
0,
        -4.5000e+00, -4.4000e+00, -4.3000e+00, -4.2000e+00, -4.1000e+0
0,
        -4.0000e+00, -3.9000e+00, -3.8000e+00, -3.7000e+00, -3.6000e+0
0,
        -3.5000e+00, -3.4000e+00, -3.3000e+00, -3.2000e+00, -3.1000e+0
0,
        -3.0000e+00, -2.9000e+00, -2.8000e+00, -2.7000e+00, -2.6000e+0
0,
        -2.5000e+00, -2.4000e+00, -2.3000e+00, -2.2000e+00, -2.1000e+0
0,
        -2.0000e+00, -1.9000e+00, -1.8000e+00, -1.7000e+00, -1.6000e+0
0,
        -1.5000e+00, -1.4000e+00, -1.3000e+00, -1.2000e+00, -1.1000e+0
0,
        -1.0000e+00, -9.0000e-01, -8.0000e-01, -7.0000e-01, -6.0000e-0
1,
        -5.0000e-01, -4.0000e-01, -3.0000e-01, -2.0000e-01, -1.0000e-0
1,
        -5.9605e-09,  1.0000e-01,  2.0000e-01,  3.0000e-01,  4.0000e-0
1,
         5.0000e-01,  6.0000e-01,  7.0000e-01,  8.0000e-01,  9.0000e-0
1,
         1.0000e+00,  1.1000e+00,  1.2000e+00,  1.3000e+00,  1.4000e+0
0,
         1.5000e+00,  1.6000e+00,  1.7000e+00,  1.8000e+00,  1.9000e+0
0,
         2.0000e+00,  2.1000e+00,  2.2000e+00,  2.3000e+00,  2.4000e+0
0,
         2.5000e+00,  2.6000e+00,  2.7000e+00,  2.8000e+00,  2.9000e+0
0,
         3.0000e+00,  3.1000e+00,  3.2000e+00,  3.3000e+00,  3.4000e+0
0,
         3.5000e+00,  3.6000e+00,  3.7000e+00,  3.8000e+00,  3.9000e+0
0,
         4.0000e+00,  4.1000e+00,  4.2000e+00,  4.3000e+00,  4.4000e+0
```

```
0,
         4.5000e+00,  4.6000e+00,  4.7000e+00,  4.8000e+00,  4.9000e+0
0,
         5.0000e+00,  5.1000e+00,  5.2000e+00,  5.3000e+00,  5.4000e+0
0,
         5.5000e+00,  5.6000e+00,  5.7000e+00,  5.8000e+00,  5.9000e+0
0])
tensor([0.0025, 0.0027, 0.0030, 0.0033, 0.0037, 0.0041, 0.0045, 0.005
0, 0.0055,
        0.0061, 0.0067, 0.0074, 0.0082, 0.0090, 0.0100, 0.0110, 0.012
1, 0.0134,
        0.0148, 0.0163, 0.0180, 0.0198, 0.0219, 0.0241, 0.0266, 0.029
3, 0.0323,
        0.0356, 0.0392, 0.0431, 0.0474, 0.0522, 0.0573, 0.0630, 0.069
1, 0.0759,
        0.0832, 0.0911, 0.0998, 0.1091, 0.1192, 0.1301, 0.1419, 0.154
5, 0.1680,
        0.1824, 0.1978, 0.2142, 0.2315, 0.2497, 0.2689, 0.2891, 0.310
0, 0.3318,
        0.3543, 0.3775, 0.4013, 0.4256, 0.4502, 0.4750, 0.5000, 0.525
0, 0.5498,
        0.5744, 0.5987, 0.6225, 0.6457, 0.6682, 0.6900, 0.7109, 0.731
1, 0.7503,
        0.7685, 0.7858, 0.8022, 0.8176, 0.8320, 0.8455, 0.8581, 0.869
9, 0.8808,
        0.8909, 0.9002, 0.9089, 0.9168, 0.9241, 0.9309, 0.9370, 0.942
7, 0.9478,
        0.9526, 0.9569, 0.9608, 0.9644, 0.9677, 0.9707, 0.9734, 0.975
9, 0.9781,
        0.9802, 0.9820, 0.9837, 0.9852, 0.9866, 0.9879, 0.9890, 0.990
0, 0.9910,
        0.9918, 0.9926, 0.9933, 0.9939, 0.9945, 0.9950, 0.9955, 0.995
9, 0.9963,
        0.9967, 0.9970, 0.9973])

<Figure size 640x480 with 1 Axes>
```

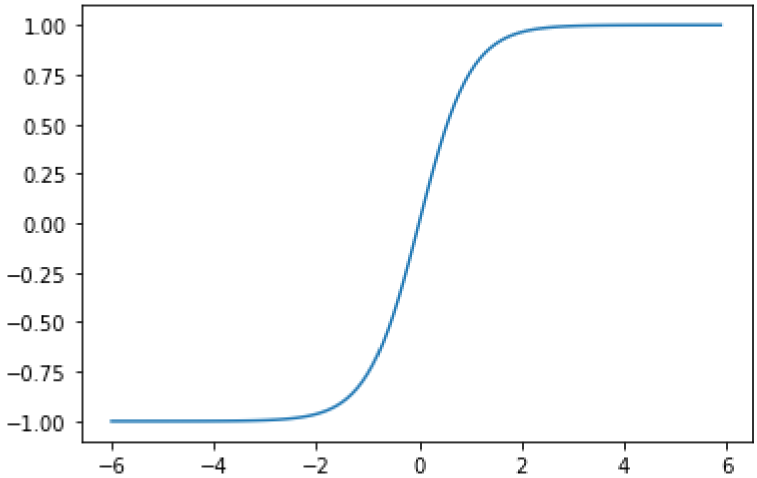
Tanh

It is a variant of the sigmoid function, that maps a set of real values to the range between -1 and +1

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

In [28]:

```
y = torch.tanh(x)
# print(y)
plt.plot(x.numpy(), y.detach().numpy())
plt.show()
```



ReLU

It stands for `rectified linear unit` and it is the most important activation function. It clips the negative values to 0

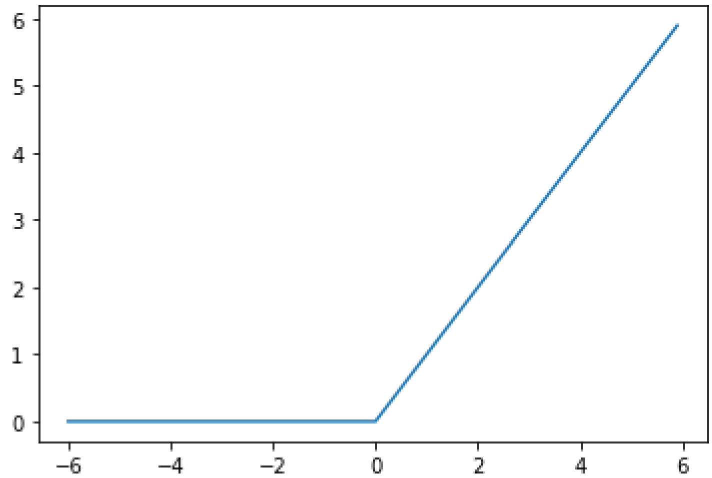
$$f(x) = max(0, x)$$

To avoid the issue of `dying ReLU` where certain outputs become zero and never change value, the `parametric ReLU` is proposed, where a leak coefficient `a` is a learned parameter.

$$f(x) = max(x, ax)$$

In [29]:

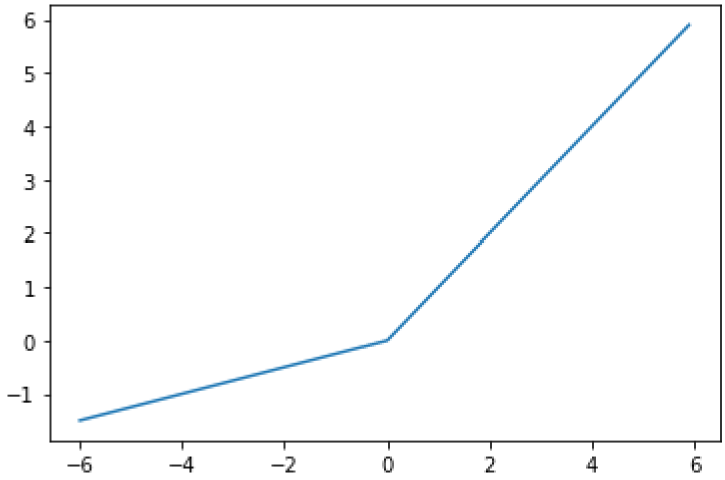
```
y = torch.relu(x)
# print(y)
plt.plot(x.numpy(),  y.detach().numpy())
plt.show()
```



In [30]:

```

prelu = nn.PReLU(num_parameters=1)
y = prelu(x)
plt.plot(x.numpy(), y.detach().numpy())
plt.show()
```



Softmax

It is like the sigmoid function, where the output of each unit are between 0 and 1, while it divides the output by the discrete probability distribution (value between 0 and 1 and sum of each output is 1)

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$

In [31]:

```python
torch.manual_seed(0) # setting a seed to reproduce
softmax = nn.Softmax(dim=1)
x = torch.randn(1, 3)
y = softmax(x)
print("input:", x)
print("softmax output:",y)
print("distribution:", torch.sum(y, dim=1))
```

```
input: tensor([[ 1.5410, -0.2934, -2.1788]])
softmax output: tensor([[0.8446, 0.1349, 0.0205]])
distribution: tensor([1.])
```

Loss Functtions

A loss function takes the truth value (y) and a prediction ($\hat{y}$) as an input and produce a real-valued score. The lower this score, the better the model is. Some of the most loss functions in PyTorch are the following.

Mean Squared Error Loss

When the out put (`y`) and the prediction ($\hat{y}$) are continous values, we can use mean squared error (`MSE`).

$$L_{MSE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y - \hat{y})^2$$

In [32]:

```
mse_loss = nn.MSELoss()
#predictions
outputs = torch.tensor([2,3], dtype=torch.float, requires_grad=True)
print("Outpusts-->", outputs)

#ground truth
targets = torch.tensor([1.5, 2.5],dtype=torch.float)
print("Targets-->", targets)
loss = mse_loss(outputs, targets)
loss.backward()
print("loss-->",loss)
```

```
Outpusts--> tensor([2., 3.], requires_grad=True)
Targets--> tensor([1.5000, 2.5000])
loss--> tensor(0.2500, grad_fn=<MseLossBackward0>)
```

Categorical Cross-Entropy Loss

Categorical cross-entropy is a loss function that is used in multi-class classification tasks. The categorical cross-entropy loss function calculates the loss of an example by computing the following sum:

$$L_{CE}(y, \hat{y}) = -\sum_{i=1}^{n} y_i . log(\hat{y})$$

In [33]:

```
ce_loss = nn.CrossEntropyLoss()
outputs = torch.tensor([[1,2,3,4],[3,4, 5,6],[1,5, 8,9]], dtype=torch.float, req

print ("outputs-->",outputs)
targets = torch.tensor([1, 0, 3], dtype=torch.int64)
loss = ce_loss(outputs, targets)
loss.backward()
print ("Losses -->", loss)
```

```
outputs--> tensor([[1., 2., 3., 4.],
        [3., 4., 5., 6.],
        [1., 5., 8., 9.]], requires_grad=True)
Losses --> tensor(2.0691, grad_fn=<NllLossBackward0>)
```

[Binary Cross-Entropy Loss (https://towardsdatascience.com/unders binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a)](https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a)

In [34]:

```
bce_loss = nn.BCELoss()
sigmoid = nn.Sigmoid()
probabilities = sigmoid(torch.randn(4, 1, requires_grad=True))
print(probabilities)

targets = torch.tensor([1, 0, 1, 0], dtype=torch.float32).view(4, 1)
loss = bce_loss(probabilities, targets)
loss.backward()
print(loss)
```

```
tensor([[0.6384],
        [0.2527],
        [0.1980],
        [0.5995]], grad_fn=<SigmoidBackward0>)
tensor(0.8186, grad_fn=<BinaryCrossEntropyBackward0>)
```

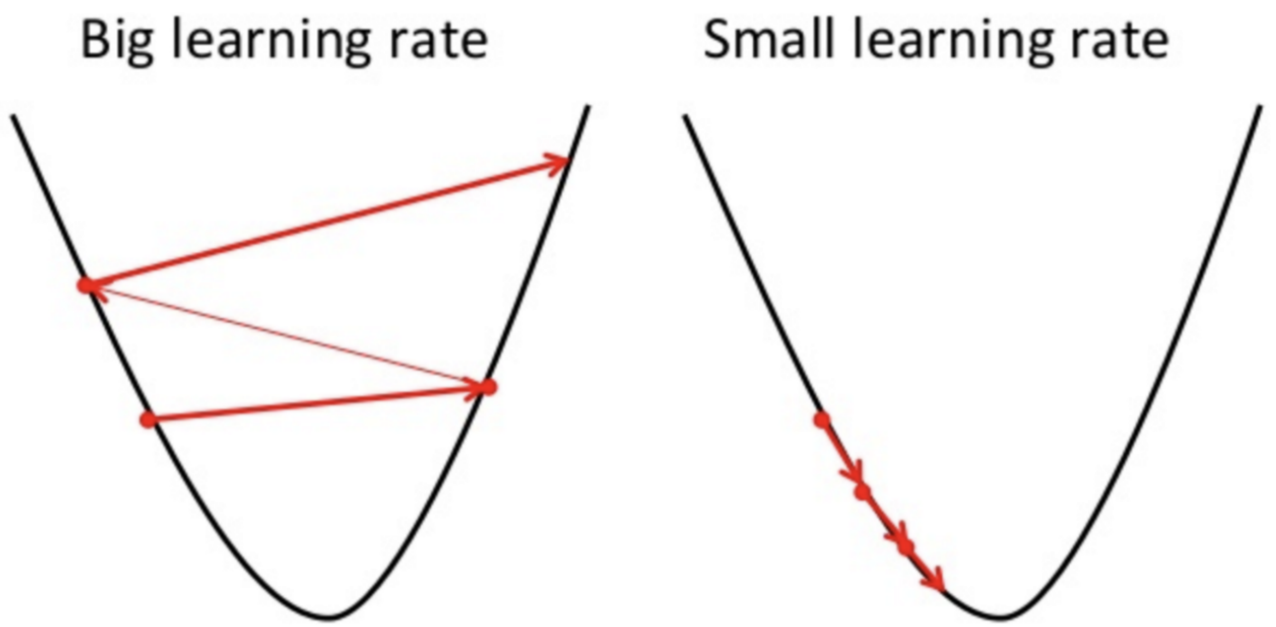
Optimizers

The optimizer tie together the loss function and model parameters by updating the model in response to the output of the loss function. The following are the main steps for optimizers:

1. Calculate what a small change in each individual weight would do to the loss function

2. Adjust each individual weight based on its gradient (i.e. take a small step in the determined direction)
3. Keep doing steps #1 and #2 until the loss function gets as low as possible

Learning rates can help to regularize the change weights for a better convergence.



Optimizers in PyTorch

- Adam
- Adagrad
- RMSprop

- SGD ...

Epochs and Batches

During training, we need to update the parameters in iterations. An iteration in neural network training is one parameter update step. That is, in each iteration, each parameter is updated once. To do so, we divide the training data into different min-batches.

An epoch is a measure of the number of times all training data is used once to update the parameters.

Supervised Model in PyTorch

Recaps: a supervised approach needs a `model`, a `loss function`, `training data`, and an `optimization algorithm`.

Example : Text Categorization - [The 20 Newsgroups data set (http://qwone.com/~jason/20New](http://qwone.com/~jason/20Newsgroups/)

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.

In [35]:

```python
import torch
import pandas as pd
import numpy as np
import sklearn
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
```

In [36]:

```
# get sub categories
categories = ["comp.graphics","sci.space","rec.sport.baseball","talk.politics.gu
newsgroups_train = fetch_20newsgroups(subset='train', categories=categories)
newsgroups_test = fetch_20newsgroups(subset='test', categories=categories)

print('total texts in train:',len(newsgroups_train.data))
print('total texts in test:',len(newsgroups_test.data))
```

```
total texts in train: 2919
total texts in test: 1942
```

In [37]:

```
# Getting all the vocabularies and indexing to a unique position
vocab = Counter()
#Indexing words from the training data
for text in newsgroups_train.data:
    for word in text.split(' '):
        vocab[word.lower()]+=1

#Indexing words from the test data
for text in newsgroups_test.data:
    for word in text.split(' '):
        vocab[word.lower()]+=1

total_words = len(vocab)

def get_word_2_index(vocab):
    word2index = {}
    for i,word in enumerate(vocab):
        word2index[word.lower()] = i

    return word2index

word2index = get_word_2_index(vocab)
```

In [38]:

```
print(len(word2index))
print(word2index["the"]) # Showing the index of 'the'
print (total_words)
```

```
196609
72
196609
```

In [39]:

```
def get_batch(df,i,batch_size):
    batches = []
    results = []
    # Split into different batchs, get the next batch
    texts = df.data[i*batch_size:i*batch_size+batch_size]
    # get the targets
    categories = df.target[i*batch_size:i*batch_size+batch_size]
    for text in texts:
        # Dimension, 196609
        layer = np.zeros(total_words,dtype=float)

        for word in text.split(' '):
            layer[word2index[word.lower()]] += 1
        batches.append(layer)

    # We have 5 categories
    for category in categories:
        index_y = -1
        if category == 0:
            index_y = 0
        elif category == 1:
            index_y = 1
        elif category == 2:
            index_y = 2
        elif category == 3:
            index_y = 3
        else:
            index_y = 4
        results.append(index_y)

    # the training and the targets
    return np.array(batches),np.array(results)
```

In [40]:

```
# Parameters
learning_rate = 0.01
num_epochs = 10
batch_size = 150
display_step = 1

# Network Parameters
hidden_size = 100      # 1st layer and 2nd layer number of features
input_size = total_words # Words in vocab
num_classes = 5         # Categories: "graphics","space","baseball","guns", "chr
```

In [41]:

```
import torch.nn as nn
```

In [42]:

```
# define the network
class News_20_Net(nn.Module):
     def __init__(self, input_size, hidden_size, num_classes):
        super(News_20_Net, self).__init__()
        self.layer_1 = nn.Linear(input_size,hidden_size, bias=True)
        self.relu = nn.ReLU()
        self.layer_2 = nn.Linear(hidden_size, hidden_size, bias=True)
        self.output_layer = nn.Linear(hidden_size, num_classes, bias=True)
    # accept input and return an output
     def forward(self, x):
        out = self.layer_1(x)
        out = self.relu(out)
        out = self.layer_2(out)
        out = self.relu(out)
        out = self.output_layer(out)
        return out
```

In [43]:

```python
news_net = News_20_Net(input_size, hidden_size, num_classes)
# Loss and Optimizer
criterion = nn.CrossEntropyLoss()  # This includes the Softmax loss function
optimizer = torch.optim.Adam(news_net.parameters(), lr=learning_rate)

# Train the Model
for epoch in range(num_epochs):
    # determine the number of min-batches based on the batch size and size of tr
    total_batch = int(len(newsgroups_train.data)/batch_size)
    # Loop over all batches
    for i in range(total_batch):
        batch_x,batch_y = get_batch(newsgroups_train,i,batch_size)
        articles = torch.FloatTensor(batch_x)
        labels = torch.LongTensor(batch_y)
        #print("articles",articles)
        #print(batch_x, labels)
        #print("size labels",labels.size())

        # Forward + Backward + Optimize
        optimizer.zero_grad()  # zero the gradient buffer
        outputs = news_net(articles)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        if (i+1) % 4 == 0:
            print ('Epoch [%d/%d], Step [%d/%d], Loss: %.4f'
                   %(epoch+1, num_epochs, i+1,
                     len(newsgroups_train.data)/batch_size, loss.data))
```

```
Epoch [1/10], Step [4/19], Loss: 1.4060
Epoch [1/10], Step [8/19], Loss: 0.7800
Epoch [1/10], Step [12/19], Loss: 0.3270
Epoch [1/10], Step [16/19], Loss: 0.3120
Epoch [2/10], Step [4/19], Loss: 0.0227
Epoch [2/10], Step [8/19], Loss: 0.0078
Epoch [2/10], Step [12/19], Loss: 0.1025
Epoch [2/10], Step [16/19], Loss: 0.1227
```

```
Epoch [3/10], Step [4/19], Loss: 0.0486
Epoch [3/10], Step [8/19], Loss: 0.0059
Epoch [3/10], Step [12/19], Loss: 0.0167
Epoch [3/10], Step [16/19], Loss: 0.2845
Epoch [4/10], Step [4/19], Loss: 0.0020
Epoch [4/10], Step [8/19], Loss: 0.0085
Epoch [4/10], Step [12/19], Loss: 0.0160
Epoch [4/10], Step [16/19], Loss: 0.0417
Epoch [5/10], Step [4/19], Loss: 0.0001
Epoch [5/10], Step [8/19], Loss: 0.0003
Epoch [5/10], Step [12/19], Loss: 0.0419
```

In [44]:

```
#show the different trained parameters
for name, param in news_net.named_parameters():
    if param.requires_grad:
        print ("Name--->",name,"\nValues--->", param.data)
```

```
Name---> layer_1.weight
Values---> tensor([[ 1.0801e-01,  6.3735e-02,  6.4451e-02,  ..., -1.2
464e-03,
         -1.2879e-03,  4.6998e-05],
        [ 1.0809e-01, -6.1793e-02, -1.1131e-01,  ...,  7.6643e-04,
         -6.4708e-04,  3.1182e-04],
        [ 1.4574e-01,  6.1897e-02,  6.1568e-02,  ...,  1.7765e-03,
          1.4413e-03, -8.8624e-04],
        ...,
        [-5.1072e-03, -1.6860e-04, -2.0744e-03,  ...,  1.1594e-03,
         -8.7936e-04,  1.9404e-03],
        [-6.0090e-02,  8.8442e-04,  1.3485e-03,  ..., -1.8278e-03,
          9.4753e-04,  1.2488e-03],
        [ 1.3802e-01,  5.5154e-02,  2.3579e-02,  ...,  1.6811e-03,
         -7.9258e-04,  1.3332e-03]])
Name---> layer_1.bias
Values---> tensor([ 0.0790,  0.1035,  0.1286, -0.0576,  0.0730,  0.11
40,  0.1691, -0.0505,
        -0.0715,  0.1311,  0.2025, -0.0276,  0.0812, -0.0246,  0.099
8,  0.1465
```

In [45]:

```
# Test the Model
correct = 0
total = 0
total_test_data = len(newsgroups_test.target)
# get all the test dataset and test them
batch_x_test,batch_y_test = get_batch(newsgroups_test,0,total_test_data)
articles = torch.FloatTensor(batch_x_test)
labels = torch.LongTensor(batch_y_test)
outputs = news_net(articles)
_, predicted = torch.max(outputs.data, 1)
total += labels.size(0)
correct += (predicted == labels).sum()
print('Accuracy of the network on the 1180 test articles: %d %%' % (100 * correc
```

```
Accuracy of the network on the 1180 test articles: 91 %

/Users/abhikjana/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:13: UserWarning: __floordiv__ is deprecated, and its behavior w
ill change in a future version of pytorch. It currently rounds toward
0 (like the 'trunc' function NOT 'floor'). This results in incorrect r
ounding for negative values. To keep the current behavior, use torch.d
iv(a, b, rounding_mode='trunc'), or for actual floor division, use tor
ch.div(a, b, rounding_mode='floor').
  del sys.path[0]
```

ML Assignments - 50 points

You need to choose one option. Project can be done in group (max 4 students). There will be presentation 6.12/7.12.2022. You need to prepare a small report (1-2 pages) and a presentation (max 5 minutes). All of the group members should participate in the presentation.

Assignment option One - Train Spacy NER model using the GermaNER dataset

In this assignment your tasks are the following

1. Train a new spaCy NER model for German (blank) using the the GermaNER datasets - see Assignment-Otion-1 folder

2. Load the existing German NER model from spaCy and update the model using the GermaNER dataset. Make sure to normalize entity types accordingly. For example, if the builtin entity type for person is PERSON but it is marked as PER in the GermaNER dataset, convert the GermaNER lable to PERSON. There are also some special "derivative" and "part" lables such as ORGpart and ORGderiv. How is the performance if you normalize these types to a common one, example ORGpart and ORGderiv to ORG?
3. Train and update the model using the train and dev dataset and test it with test dataset. Use the perl script provided to evaluate the performance.
4. The test file will be provided on Dec. 4, 2022

   Note: You can ignore the last column from training and testing! For the evaluation purpose, you can re-use the gold labels.

The following snippet shows an example of the TSV format we use in this task.

1 Aufgrund O O

2 seiner O O

3 Initiative O O

4 fand O O

5 2001/2002 O O

6 in O O

7 Stuttgart B-LOC O

8 , O O

9 Braunschweig B-LOC O

10 und O O

11 Bonn B-LOC O

12 eine O O

13 große O O

14 und O O

15 publizistisch O O

16 vielbeachtete O O

17 Troia-Ausstellung B-LOCpart Ov 18 statt O O

19 , O O

20 „ O O

21 Troia B-OTH B-LOC

22 - I-OTH O

23 Traum I-OTH O

24 und I-OTH O

25 Wirklichkeit I-OTH O

26 “ O O

27 . O O

Note that, you have to use the provided scripts for the evaluation of your results

The Perl scripts can be executed for the NER classifier file as:

```
perl conlleval_ner.pl < your_classified_file
```

and vice versa for the chunking file as:

```
perl conlleval_chunking.pl <
your_classified_file
```

Assignment option Two - NER using Pycrfsuite

In this assignment option, your tasks are the following

1. Train a new NER or Chunking model using Pycrfsuite library using the data - see Assignment-Option-2 folder
2. The Training and development sets are provided for both the NER and chunking tasks
3. Write the outputs of each dev/test results to a file system and use the provided perl script to evaluate the performances

4. The test file will be provided on Dec. 4, 2022

The datasets for both tasks are in the CoNLL-Format. This format specifies, that each word is written at the beginning of a new line followed by a set of columns (different features/labels). In most cases, the last column is the one which will be predicted. An empty line specifies the begin of a new sentence. The chunking and the named entity feature within the dataset follow the IOB1/IOB2 format. Within the NER dataset the IOB1 and in the Chunking dataset the IOB2 format is used.

In the IOB1 format, annotations begin with an I. Successive words with equal NE annotation or chunking annotation form a phrase. The annotation starts only with a B, if two different phrases of the same type follow each other immediately.

The IOB2 format specifies, that each new phrase has to begin with a B. An equal annotation starting with I only follows if it belongs to the same phrase.

Examples: IOB tags for NER

The U.N. Organization special representative Anthony Nyakyi Person said :

word	IOB1	IOB2
The	O	O
U.N.	I-ORG	B-ORG
special	O	O
representatives	O	O
Anthony	I-PER	B-PER
Nyakyi	I-PER	I-PER

Examples: IOB tags for Chunking

He NP reckons VP the current account deficit NP

will narrow VP to PP only # 1.8 billion NP in PP September NP . O

	IOB1	IOB2
He	I-NP	B-NP
reckons	I-VP	B-VP
the	I-NP	B-NP
current	I-NP	I-NP
account	I-NP	I-NP
deficit	I-NP	I-NP
will	I-VP	B-VP
narrow	I-VP	I-VP
to	I-PP	B-PP
only	I-NP	B-NP
#	I-NP	I-NP
1.8	I-NP	I-NP
billion	I-NP	I-NP
in	I-PP	B-PP
September	I-NP	B-NP
.	O	O

Each data set has two files: A training set for learning a model from selected features and a development set to test and finetune the performance of the model.
You should avoid combining these files while you are evaluating the feature selection of your algorithm:
Train only on the training set or a subset of the training set, and test only on the development/test set.
All data sets are located on the Moodle platform.

Note that, you have to use the provided scripts for the evaluation of your results

The Perl scripts can be executed for the NER classified file via:

```
perl conlleval_ner.pl <
your_classified_file
```

and vice versa for the chunking file via:

```
perl conlleval_chunking.pl <
your_classified_file
```

Assignment option Three - Complex word Identification (CWI)

In this assignment option, your tasks are the following

1. Train a CWI model using the training and development sets
2. You can use any approach (CRF, classification with Sklearn, spaCy model from scratch, Pytorch with one-hot-encoding, PyTorch with pre-trained embeddings)
3. The test file will be provided on Dec. 4, 2022

The task is to identify complex or hard words from a given sentence based on the training examples. The data is formatted as follows:

Examples

ID1 Both China and the Philippines flexed their muscles on Wednesday. 31 37 flexed 2 7 9

ID1 Both China and the Philippines flexed their muscles on Wednesday. 31 51 flexed their muscles 4 2 6

Here, we can see that the phrase "flexed" is marked as a complex phrase by 2 native and 7 non-native English speakers whereas the phrase "flexed their muscles" is marked by 4 native and 2 non native English speakers.

Assignment option Four - News Categorization using

PyTorch

Download the dataset from https://www.kaggle.com/uciml/news-aggregator-dataset (https://www.kaggle.com/uciml/news-aggregator-dataset) and develop a news classification or categorization model. The dataset contain only titles of a news item and some metadata. The categories of the news items include one of: – b : business – t : science and technology – e : entertainment and –m : health.

1. Prepare training and test dataset: Split the data into training and test set (80% train and 20% test). Make sure they are balanced, otherwise if all b files are on training, your model fails to predict t files in test.

2. Binary classification: produce training data for each two categories, such as b and t, b and m, e and t and so on. Evaluate the performance and report which categories are easier for the models.
3. Adapt the Text Categorization PyTorch code (see above) and evaluate the performance of the system for these task
4. Use a pre-trained embeddings and compare your result. When you use pre-ttrained embeddings, you have to average the word embeddings of each tokens in ach document to get the unique representation of the document. DOC_EMBEDDING = (TOKEN1_EMBEDDING + ... + TOKENn_EMBEDDING). You can also use some of the spacy/FLAIR document embedding methods
5. Report the recall, precision, and F1 scores for both binary and multi-class classification.

Documentation

Write a short documentation about your component. Write about the different models, algorithms, features you have developed and document the evaluation for the classification based on the development set with different features and their combination. Please also write about features you used, even if they did not improve the result.

Presentation

There will be a presentation on 6.12/7.12. 2022. Please prepare some slides to present your results within 5 minutes. Report about your evaluation, your chosen features and the best performance of your classification component.

Project Choice

You have time until Thursday 24/11/2022, 10:00 to form a group. On Thursday 10:00, we will enable to choose project options (Options 1-4). ONLY ONE student from the group should choose the project option.

- ○ Assignment Option One - Train Spacy NER model using the GermaNER dataset
  Responses: 0
  Limit: 6
- ○ Assignment Option Two - NER using Pycrfsuite
  Responses: 0
  Limit: 6
- ○ Assignment Option Three - Complex word Identification
  Responses: 0
  Limit: 6
- ○ Assignment Option Four - News Categorization using PyTorch
  Responses: 0
  Limit: 8

BONUS - Running Flask to have

classifier API

You can load and run the classifier models using Flask and expose the Predict method as a REST-API so that you can use the classifiers from a web-page, mobile devices,...

See the example app.py. This code loads the spacy English model and serve NER services.

Start the server as: `python app.py`

Once the server is started, you can test the application from the browser: https://localhost:5001/ (https://localhost:5001/)

Note you can change the port in the app.py file

Replace the spaCy model with your own classifier and improve the UI (see templates.html file)

Resources

- Pytorch Tutorial (https://speech.ee.ntu.edu.tw/~hylee/ml/ml2022-course-data/Pytorch%20Tutorial%201.pdf)
- Custom NER using spaCy (https://medium.com/@manivannan_data/how-to-train-ner-with-custom-training-data-using-spacy-188e0e508c6)
- Custom NER using spaCy (https://github.com/Jcharis/Natural-Language-Processing-Tutorials/blob/master/Training%20the%20Named%
- Overview of PyTorch (https://www.learnopencv.com/pytorch-for-beginners-basics/)
- Perceptron in PyTorch (https://medium.com/@tomgrek/building-your-

first-neural-net-from-scratch-with-pytorch-56b0e9c84d54)

- Data Loading and processing in PyTorch (https://pytorch.org/tutorials/beginner/data_loading
- PyTorch resources (https://github.com/ritchieng/the-incredible-pytorch)
- PyTorch Tutorial (https://github.com/yunjey/pytorch-tutorial)
- A Beginner-Friendly Guide to PyTorch and How it Works from Scratch (https://medium.com/analytics-vidhya/a-beginner-friendly-guide-to-pytorch-and-how-it-works-from-scratch-25c7c2dceb30)
- RNN Tutorial (https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/)

- [PyTorch Tutorial (https://pythonprogramming.net/introduction-deep-learning-neural-network-pytorch/)](https://pythonprogramming.net/introduction-deep-learning-neural-network-pytorch/)
- [NER API Flask (https://github.com/susanli2016/Named-Entity-Extractor)](https://github.com/susanli2016/Named-Entity-Extractor)

In [ ]:

```
 
```