# 1) Problem statment

Companies face a lot of customer requests every working day, which may include installation, maintenance, complaints, and troubleshooting of a particular product. Those requests come in form of unstructured text data. To provide a valuable service, the companies need to employ many people to fastly read, understand and process those requests as soon as possible to stay competitive in the markets. To speed up the processing delay, we can automatize some of those processes. Then, our goal is to build an automated method to extract meaningful information from raw text data having a large number of transaction descriptions.

# 2) Algorithms used to Approch the problem

The Named Entity Recognition algorithm from spacy will be used to automize the process to extract a meaningfull number inside a transaction description.

# 3) Data description

There are three explanatory variables in the dataset: transaction descriptor, store number, and dataset with three levels (training, validation, and testing)

# 4) Experimental Procedure

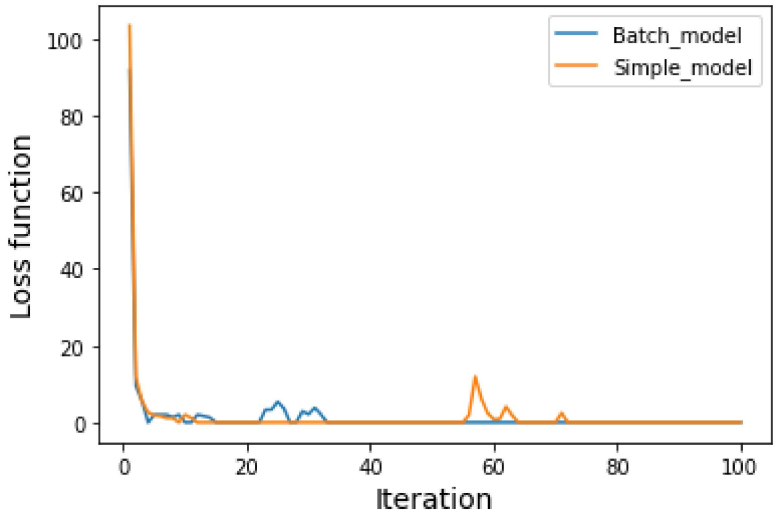
In the experimental procedure, we train two models. In the first case, we start with a blank NER model, while in the second case, we start with a pre-train model.

## a) Data pre-processing

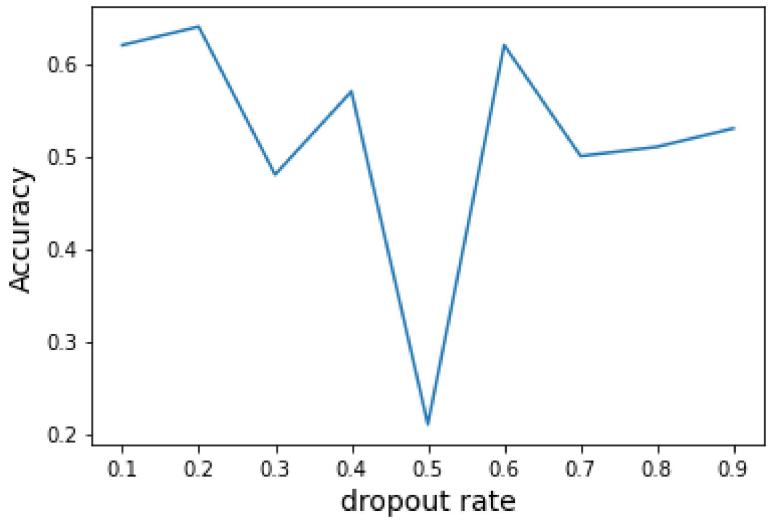
The following step were consider : remove all the special characters , lowercase the letters, delete the unnecessary white spaces and transform the training data to a tuple in which the first element contains the transaction descriptor, and the second element contains the label, starting and ending index of the number to extract inside the transaction descriptor.

## b) Implementation

We have trained the models using the same data and dropout rate, but the blank model have been trained using the batch. We then compare both models using their loss function see the figure below.




The figure show that the best model is the pre-train model (simple model). We then use the best model to tune the hyperparameter dropout rate which is the way to optimizer a neural network. We use the grid search and the prediction accuracy to select the best dropout rate value $0.2$ with $64\%$ as maximun value of accuracy on test data (see the figure below). The recommended value is $0.2$ which means that about $20\%$ of the neurons used in this model will be dropped randomly during training.




## c) Model entity extraction on test data

We have trained the best model and archived $66\%$ test accuracy. Our model mostly fail to extract number from the following format ut044, f35869mcdonalds.

# 5) Code use to build the NER

In [ ]:

```
# Load Pkgs
import pandas as pd
import numpy as np
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
# Load NLP Pkgs
import spacy
from wordcloud import WordCloud, STOPWORDS
from spacy.util import minibatch, compounding
import matplotlib.pyplot as plt
import re
import random
from spacy.training.example import Example
```

In [3]:

```
# Load Dataset
df = pd.read_csv("Summer Internship - Homework Exercise.csv",encoding = "UTF-8")
len(df)
```

Out[3]:

```
300
```

In [4]:

```
df.head()
```

Out[4]:

| | transaction_descriptor | store_number | dataset |
|---|---|---|---|
| **0** | DOLRTREE 2257 00022574 ROSWELL | 2257 | train |
| **1** | AUTOZONE #3547 | 3547 | train |
| **2** | TGI FRIDAYS 1485 0000 | 1485 | train |
| **3** | BUFFALO WILD WINGS 003 | 3 | train |
| **4** | J. CREW #568 0 | 568 | train |

# Remove special character and punctuation

In [5]:

```
replacer = {'\n':'',"[\[].*?[\]]": "",'[!"#%\'()*+,-./:;<=>?@\[\]^_`{|}~’""’‘\\\]':"", ' +'
df['transaction_descriptor'] = df['transaction_descriptor'].replace(replacer, regex=True)
df.head()
```

Out[5]:

| | transaction_descriptor | store_number | dataset |
|---|---|---|---|
| **0** | DOLRTREE 2257 00022574 ROSWELL | 2257 | train |
| **1** | AUTOZONE 3547 | 3547 | train |
| **2** | TGI FRIDAYS 1485 0000 | 1485 | train |
| **3** | BUFFALO WILD WINGS 003 | 3 | train |
| **4** | J CREW 568 0 | 568 | train |

In [6]:

```
# NER
nlp0 = spacy.load('en_core_web_sm')
```

In [7]:

```
# Get All Components of this NLP Object
nlp0.pipe_names
```

Out[7]:

```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

In [8]:

```
ner0 = nlp0.get_pipe('ner')
```

## Preparing the data

Training data must be a tuple

TRAIN_DATA = [ ("Who is Shaka Khan?", {"entities": [(START, STOP, "LABEL")]}) ]

TRAIN_DATA = [ ("Who is Shaka Khan?", {"entities": [(7, 17, "PERSON")]}), ("I like London and Berlin.", {"entities": [(7, 13, "LOC"), (18, 24, "LOC")]}), ]

In [14]:

```
def process_review(review):
    processed_token = []
    for token in review.split():
        token = ''.join(e.lower() for e in token if e.isalnum())
        processed_token.append(token)
    return ' '.join(processed_token)
```

In [15]:

```
df['transaction_descriptor']
```

Out[15]:

```
0      DOLRTREE 2257 00022574 ROSWELL
1                         AUTOZONE 3547
2                 TGI FRIDAYS 1485 0000
3               BUFFALO WILD WINGS 003
4                          J CREW 568 0
                    ...
295                    MCDONALDS F2151
296           NST BEST BUY 1403 332411
297                  CVSPHARMACY 06689
298               BANANA REPUBLIC 8109
299                 BOSTON MARKET 0443
Name: transaction_descriptor, Length: 300, dtype: object
```

# Let's split the data into train and test data.

In [16]:

```
train = df[df["dataset"]!="test" ]
len(train)
```

Out[16]:

```
200
```

In [18]:

```
test = df[df["dataset"]=="test"]
len(val)
```

Out[18]:

```
100
```

In [19]:

```
count = 0
TRAIN_DATA = []
TEST_DATA = []
for _, item in train.iterrows():
    ent_dict = {}
    if count < 1000:
        review = process_review(item['transaction_descriptor'])
        #Locate drugs and their positions once and add to the visited items.
        visited_items = []
        entities = []
        for token in review.split():
            if token in all_drugs:
                for i in re.finditer(token, review):
                    if token not in visited_items:
                        entity = (i.span()[0], i.span()[1], 'store_number')
                        visited_items.append(token)
                        entities.append(entity)
        if len(entities) > 0:
            ent_dict['entities'] = entities
            train_item = (review, ent_dict)
            TRAIN_DATA.append(train_item)
            count+=1

for _, item in test.iterrows():
    ent_dict = {}
    if count < 1000:
        review = process_review(item['transaction_descriptor'])
        #Locate drugs and their positions once and add to the visited items.
        visited_items = []
        entities = []
        for token in review.split():
            if token in all_drugs:
                for i in re.finditer(token, review):
                    if token not in visited_items:
                        entity = (i.span()[0], i.span()[1], 'store_number')
                        visited_items.append(token)
                        entities.append(entity)
        if len(entities) > 0:
            ent_dict['entities'] = entities
            train_item = (review, ent_dict)
            TEST_DATA.append(train_item)
            count+=1
```

In [107]:

```
TRAIN_DATA[:5]
```

Out[107]:

```
[('jamba juice 1305', {'entities': [(12, 16, 'store_number')]}),
 ('nst best buy 380 830230', {'entities': [(13, 16, 'store_number')]}),
 ('five guys mn 1847 ecom 6123399733 mn',
  {'entities': [(13, 17, 'store_number')]}),
 ('subway 26824', {'entities': [(7, 12, 'store_number')]}),
 ('wendys 514 tampa fl 7271', {'entities': [(7, 10, 'store_number')]})]
```

In [108]:

```
TEST_DATA[:5]
```

Out[108]:

```
[('innout burger 242', {'entities': [(14, 17, 'store_number')]}),
 ('jcpenney 1419', {'entities': [(9, 13, 'store_number')]}),
 ('ross stores 1019', {'entities': [(12, 16, 'store_number')]}),
 ('wm supercenter 38', {'entities': [(15, 17, 'store_number')]}),
 ('ihop 629 white house tn', {'entities': [(5, 8, 'store_number')]})]
```

# Training the NER Model

# Model 1

In [49]:

```
n_iter = 100

def train_ner(training_data):
    """Steps
    Create a Blank NLP  model object
    Create and add NER to the NLP model
    Add Labels from your training data
    Train
    """
    loss=np.array([])
    TRAIN_DATA = training_data
    nlp = spacy.blank("en")  # create blank Language class
    print("Created blank 'en' model")

    if "ner" not in nlp.pipe_names:
        ner = nlp.add_pipe("ner",last=True)
        #nlp.add_pipe(ner, last=True)
    # otherwise, get it so we can add labels
    else:
        ner = nlp.get_pipe("ner")

    # add labels
    for _, annotations in TRAIN_DATA:
        for ent in annotations.get("entities"):
            ner.add_label(ent[2])

    nlp.begin_training()
    for itn in range(n_iter):
        random.shuffle(TRAIN_DATA)
        losses = {}
        # batch up the examples using spaCy's minibatch
        batches = minibatch(TRAIN_DATA, size=compounding(4.0, 32.0, 1.001))
        for batch in batches:
            texts, annotations = zip(*batch)
            n_samples = len(texts)
            for sample, label in zip(texts[:n_samples],annotations[:n_samples]):

                    example = Example.from_dict(nlp.make_doc(sample), label)
                    nlp.update([example],
                        drop=0.2,  # dropout - make it harder to memorise data
                        losses=losses,
                    )
        print("Losses", losses)
        loss = np.append(loss,losses["ner"])
    return nlp,loss
```

dropout rate is a way to optimize a neural network.

The recommended value is 0.2 which means that about 20% of the neurons used in this model will be dropped randomly during training.

In [50]:

```
# Let training
nlp2,loss = train_ner(TRAIN_DATA)
```

```
Losses {'ner': 2.0031323514173263}
Losses {'ner': 0.000239994461452747}
Losses {'ner': 0.011907318319161347}
Losses {'ner': 1.9915631243634742}
Losses {'ner': 1.685748632409193}
Losses {'ner': 1.3018246415169326}
Losses {'ner': 0.00010720328365293833}
Losses {'ner': 7.764343654376074e-08}
Losses {'ner': 6.675830024300234e-09}
Losses {'ner': 1.8460710761405085e-10}
Losses {'ner': 0.00030318485542707695}
Losses {'ner': 2.531044831744727e-11}
Losses {'ner': 1.1237773431302094e-09}
Losses {'ner': 3.0886564637193946e-10}
Losses {'ner': 3.3059035300092803}
Losses {'ner': 3.3530338339007977}
Losses {'ner': 5.377155977101957}
Losses {'ner': 3.680516075070513}
Losses {'ner': 2.0461740370103408e-11}
Losses {'ner': 7.978067415210064e-05}
```

In [54]:

```
for text,_ in TEST_DATA[:10]:
    doc = nlp2(text)
    result = [(ent,ent.label_) for ent in doc.ents]
    print(result)
```

```
[(242, 'store_number')]
[(1419, 'store_number')]
[(1019, 'store_number')]
[(38, 'store_number')]
[(629, 'store_number')]
[(2505, 'store_number')]
[(825, 'store_number')]
[(2923, 'store_number')]
[(2039, 'store_number')]
[(382, 'store_number')]
```

In [55]:

```
TEST_DATA[:1]
```

Out[55]:

```
[('innout burger 242', {'entities': [(14, 17, 'store_number')]})]
```

# Model 2

The model 2 will be design from existing model.

In [57]:

```
nlp = spacy.load('en_core_web_sm')
n_iter=100
optimizer = nlp.begin_training()
to_train_ents = TRAIN_DATA[:100]
nlp = spacy.load('en_core_web_sm')
los=np.array([])

other_pipes = [pipe for pipe in nlp.pipe_names if pipe !='ner']

with nlp.disable_pipes(*other_pipes): # only train NER
    for itn in range(n_iter): # we are going to go throught the training data 20 times
        losses = {}
        random.shuffle(to_train_ents) # we shuffle the data
        for item in to_train_ents:
            example = Example.from_dict(nlp.make_doc(item[0]), item[1])
            nlp.update([example],
            sgd=optimizer,
            drop = 0.2,
            losses =  losses)
        #print("Ner",losses["ner"])
        print("Losses", losses)
        los = np.append(los,losses["ner"])
            # this update function Takes
                        # the model we updated at the begininig at update it with the new in
                        #about the tokens, the position of the tokens and the string.
```

```
C:\Users\bauri\anaconda3\lib\site-packages\spacy\training\iob_utils.py:141:
UserWarning: [W030] Some entities could not be aligned in the text "mcdonald
s f35869mcdonalds f35869" with entities "[(10, 16, 'store_number')]". Use `s
pacy.training.offsets_to_biluo_tags(nlp.make_doc(text), entities)` to check
the alignment. Misaligned entities ('-') will be ignored during training.
  warnings.warn(

Losses {'ner': 103.46874890236369}
Losses {'ner': 11.947277631607427}
Losses {'ner': 5.583422554334379}
Losses {'ner': 2.598222898442638}
Losses {'ner': 1.8403079622927818}
Losses {'ner': 1.6439211903942312}
Losses {'ner': 1.121429127845454}
Losses {'ner': 1.1103301682878597}
Losses {'ner': 0.022225549123086198}
Losses {'ner': 2.000058761128041}
Losses {'ner': 1.1591392948882753}
Losses {'ner': 0.00016392741093708725}
Losses {'ner': 1.738148970902224e-06}
Losses {'ner': 0.0035866812835979064}
Losses {'ner': 0.00014317396948057193}
Losses {'ner': 1.2412210455368695e-07}
Losses {'ner': 3.599590628140334e-07}
Losses {'ner': 7.956412639379859e-06}
Losses {'ner': 1.7294968780816352e-08}
Losses {'ner': 1.4804538699819264e-05}
Losses {'ner': 5.957718854260713e-10}
Losses {'ner': 2.2993613981732894e-08}
Losses {'ner': 3.285358911270288e-08}
Losses {'ner': 1.4356107836267331e-08}
```

```
Losses {'ner': 6.2054298457094406e-06}
Losses {'ner': 2.2993800807774472e-08}
Losses {'ner': 1.6310441862775436e-06}
Losses {'ner': 4.660970846471624e-08}
Losses {'ner': 9.160350980329268e-07}
Losses {'ner': 5.838903132704231e-08}
Losses {'ner': 7.065346716576813e-09}
Losses {'ner': 7.804547541792375e-09}
Losses {'ner': 2.0518676581281019e-07}
Losses {'ner': 5.915840641836702e-09}
Losses {'ner': 5.989571655465713e-09}
Losses {'ner': 1.0896983797103584e-07}
Losses {'ner': 5.230335731107021e-06}
Losses {'ner': 5.855676639156215e-09}
Losses {'ner': 6.240632885454791e-10}
Losses {'ner': 4.4662523107000433e-10}
Losses {'ner': 4.969128845160313e-10}
Losses {'ner': 3.931695310703369e-08}
Losses {'ner': 4.293313404778963e-08}
Losses {'ner': 5.095506440288943e-08}
Losses {'ner': 2.5728936322671343e-08}
Losses {'ner': 9.384545080820206e-09}
Losses {'ner': 3.4147837991039836e-10}
Losses {'ner': 3.0881035024049954e-09}
Losses {'ner': 1.2934834001324544e-08}
Losses {'ner': 1.3001539039940171e-06}
Losses {'ner': 6.696925165472889e-10}
Losses {'ner': 2.6348683913688747e-08}
Losses {'ner': 4.1738899010169647e-07}
Losses {'ner': 1.0157685832199666e-06}
Losses {'ner': 1.3870936110693727e-06}
Losses {'ner': 1.9508471530916054}
Losses {'ner': 11.988623165372028}
Losses {'ner': 6.188486081559789}
Losses {'ner': 2.4582718706686486}
Losses {'ner': 0.800867317997947}
Losses {'ner': 0.9247972886137464}
Losses {'ner': 4.10126179164289}
Losses {'ner': 1.824444861353321}
Losses {'ner': 7.761107553964034e-05}
Losses {'ner': 3.806118301885551e-10}
Losses {'ner': 1.7872488672978432e-14}
Losses {'ner': 1.0752731200521251e-14}
Losses {'ner': 9.44334335522818e-08}
Losses {'ner': 8.449781334694416e-16}
Losses {'ner': 6.656388210882961e-11}
Losses {'ner': 2.4643597310430385}
Losses {'ner': 2.768770937677836e-18}
Losses {'ner': 3.897847165573473e-14}
Losses {'ner': 1.4699844639602245e-12}
Losses {'ner': 9.039606897302627e-15}
Losses {'ner': 1.690181597036223e-05}
Losses {'ner': 1.8133545232151053e-09}
Losses {'ner': 5.3526735043010805e-14}
Losses {'ner': 1.5335763123376975e-14}
Losses {'ner': 2.12206689972247e-11}
Losses {'ner': 2.8139474274782983e-18}
Losses {'ner': 5.810421913154985e-19}
Losses {'ner': 8.576656968076609e-10}
Losses {'ner': 9.826110280241293e-17}
Losses {'ner': 3.055549463745909e-09}
```

```
Losses {'ner': 4.168863508909977e-14}
Losses {'ner': 1.456987724130261e-19}
Losses {'ner': 1.711462369102443e-18}
Losses {'ner': 1.5147934753278893e-15}
Losses {'ner': 4.824277905256844e-16}
Losses {'ner': 3.813251473254461e-19}
Losses {'ner': 8.118478978272878e-15}
Losses {'ner': 3.7386978500719076e-17}
Losses {'ner': 6.84761336448735e-17}
Losses {'ner': 2.0570631854403213e-17}
Losses {'ner': 4.994289905549624e-18}
Losses {'ner': 1.3150359418515095e-16}
Losses {'ner': 1.4777629233554936e-13}
Losses {'ner': 3.521991593641869e-14}
Losses {'ner': 5.61325988245215e-13}
```

In [58]:

```
for text,_ in TRAIN_DATA[15:20]:
    doc = nlp(text)
    result = [(ent,ent.label_) for ent in doc.ents]
    print(result)
```

```
[(2257, 'store_number')]
[(523, 'store_number')]
[(4, 'store_number')]
[(f35869mcdonalds, 'store_number')]
[(739723, 'store_number')]
```

In [59]:

```
TRAIN_DATA[1:2]
```

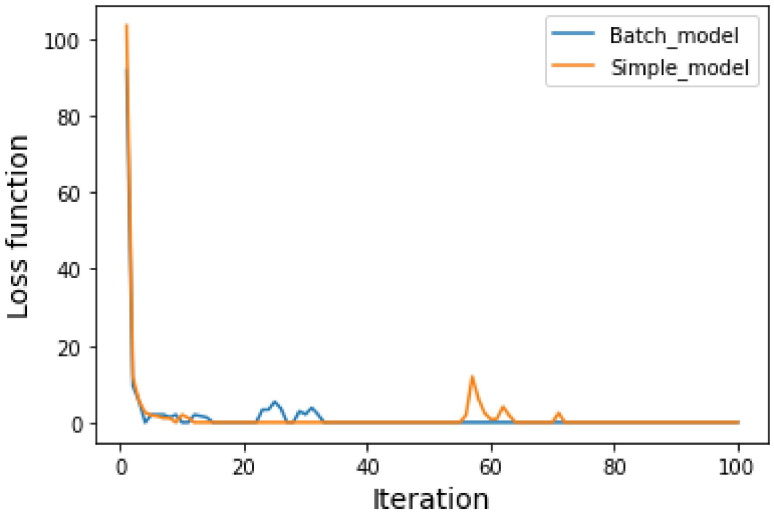
Out[59]:

```
[('nst best buy 380 830230', {'entities': [(13, 16, 'store_number')]})]
```

# Model-loss function

In [60]:

```
## plot the loss
x=np.arange(1,n_iter+1)
plt.plot(x,loss,label='Batch_model')
plt.plot(x,los,label='Simple_model')
plt.xlabel('Iteration',size=14)
plt.ylabel('Loss function',size=14)
plt.legend()
plt.show()
```




the best model is the simple model. So we will use it for the prediction

# Function to extract number from the following format : ut044, f35869mcdonalds.

In [61]:

```
def cont(inp_str):
    "This function will be useful to extract number inside the string"
    num = ""
    for c in inp_str:
        if c.isdigit():
            num = num + c
    return num
```

# Model chose for the test

In [78]:

```
def extract_drug_entity(text):
    docx =  nlp(text)
    result = [(ent,ent.label_) for ent in docx.ents]
    return result
```

In [110]:

```
test['transaction_descriptor'][:5]
```

Out[110]:

```
200          INNOUT BURGER 242
201    BP9442088LIBERTYVILLE B
202               JCPENNEY 1419
203           ROSS STORES 1019
204          WM SUPERCENTER 38
Name: transaction_descriptor, dtype: object
```

In [111]:

```
test['store_number'][:5]
```

Out[111]:

```
200        242
201    9442088
202       1419
203       1019
204         38
Name: store_number, dtype: object
```

# Entity Named Extraction : Number Extraction

In [81]:

```
l =list(test['transaction_descriptor'].apply(extract_drug_entity))
h =np.array([])
for i in range(len(l)):
    if l[i]!=[] :
        h= np.append(h,cont(str(l[i][0][0])))
    else:
        h=np.append(h,str(0))
for i in range(len(h)):
    if h[i]=='':
        h[i]='0'
h=h.astype(str)
```

# Evaluation for Spacy NER

In [83]:

```
target = test['store_number'].astype(str)
```

Sample expected (target) values

In [84]:

```
result = np.array((target ==h))
```

In [85]:

```
print("The accuracy of the model is ", sum(result)/100)
```

```
The accuracy of the model is  0.66
```

# Let's tune the selected model with different value of dropout rate

In [92]:

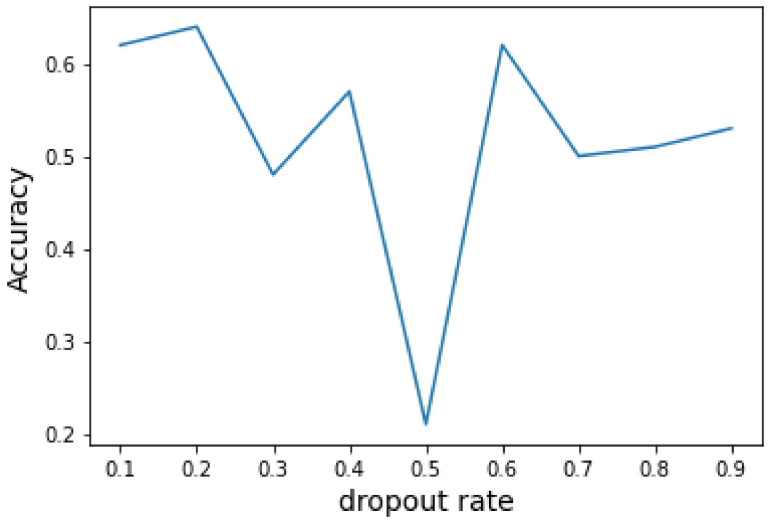
```
la = np.arange(1,10)/10
acc = np.array([])
n_iter=100
to_train_ents = TRAIN_DATA[:100]

for alpha in la:
    nlp = spacy.load('en_core_web_sm')
    optimizer = nlp.begin_training()
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe !='ner']

    with nlp.disable_pipes(*other_pipes): # only train NER
        for itn in range(n_iter): # we are going to go throught the training data 20 times
            losses = {}
            random.shuffle(to_train_ents) # we shuffle the data
            for item in to_train_ents:
                example = Example.from_dict(nlp.make_doc(item[0]), item[1])
                nlp.update([example],
                sgd=optimizer,
                drop = 0.2,
                losses =  losses)
            #print("Ner",losses["ner"])
    #         print("Losses", losses)
            los = np.append(los,losses["ner"])
                # this update function Takes
                            # the model we updated at the begininig at update it with the ne
                            #about the tokens, the position of the tokens and the string.
    l =list(test['transaction_descriptor'].apply(extract_drug_entity))
    h =np.array([])
    for i in range(len(l)):
        if l[i]!=[] :
            h= np.append(h,cont(str(l[i][0][0])))
        else:
            h=np.append(h,str(0))
    for i in range(len(h)):
        if h[i]=='':
            h[i]='0'
    h=h.astype(str)
    result = np.array((target ==h))
    acc = np.append(acc,sum(result)/100)
    print("The accuracy of the model is ", sum(result)/100 , " for", alpha)
```

```
C:\Users\bauri\anaconda3\lib\site-packages\spacy\training\iob_utils.py:141:
UserWarning: [W030] Some entities could not be aligned in the text "mcdonald
s f35869mcdonalds f35869" with entities "[(10, 16, 'store_number')]". Use `s
pacy.training.offsets_to_biluo_tags(nlp.make_doc(text), entities)` to check
the alignment. Misaligned entities ('-') will be ignored during training.
  warnings.warn(
C:\Users\bauri\anaconda3\lib\site-packages\spacy\pipeline\attributeruler.py:
150: UserWarning: [W036] The component 'matcher' does not have any patterns
defined.
  matches = self.matcher(doc, allow_missing=True, as_spans=False)
C:\Users\bauri\anaconda3\lib\site-packages\spacy\pipeline\lemmatizer.py:211:
UserWarning: [W108] The rule-based lemmatizer did not find POS annotation fo
r one or more tokens. Check that your pipeline includes components that assi
gn token.pos, typically 'tagger'+'attribute_ruler' or 'morphologizer'.
  warnings.warn(Warnings.W108)

The accuracy of the model is  0.62  for 0.1
The accuracy of the model is  0.64  for 0.2
```

```
The accuracy of the model is  0.48  for 0.3
The accuracy of the model is  0.57  for 0.4
The accuracy of the model is  0.21  for 0.5
The accuracy of the model is  0.62  for 0.6
The accuracy of the model is  0.5  for 0.7
The accuracy of the model is  0.51  for 0.8
The accuracy of the model is  0.53  for 0.9
```

In [100]:

```
## plot the loss

plt.plot(la,acc,)
plt.xlabel('dropout rate',size=14)
plt.ylabel('Accuracy',size=14)
plt.savefig('parameter.png',dpi = 100)
plt.show()
```




In [ ]: