Sayed Ahmed, Toronto, Canada
Linkedin: SayedJustetc

8112223 Canada Inc
JustEtc Social Services

CSE, BUET

BSc. Eng. in Computer Sc. & Eng.
MSc in Computer Science
MSc in Data Science and Analytics

Workplace Communication Program
Teach in Higher Education

Linkedin Learning
IBM Data Science/CognitiveAI
SkillSoft

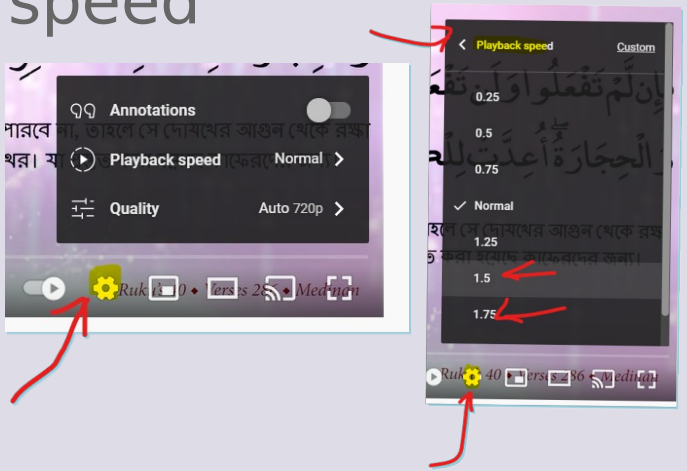
ShopForSoul.com
Training.SitesTree.com
Bangla.SaLearningSchool.com
Youtube: SaLearningSchool-ShopForSoul

NLP – Natural Language Processing

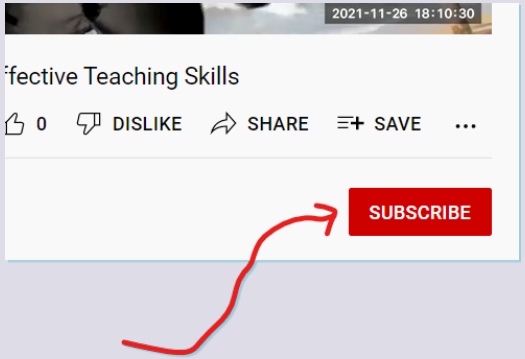
Ref: Internet Images

Youtube
Listen Faster/Slower

- Watch at 2x speed or 1.5x speed



- Subscribe



Misc

- Buy the courses
  - https://ShopForSoul.com
  - http://sitestree.com/training/
  - http://bangla.salearningschool.com/
- Get access to our LMS

- Advantages
  - Discussion
  - Chat
  - Live Sessions
  - Select topics to create videos on
  - Q & A
  - Free Courses

Learn to Install NLTK

- How to install NLTK on your local machine
- Both sets of instructions below assume you already have Python installed. These instructions are taken directly from http://www.nltk.org/install.html.
-
- Mac/Unix
-
- From the terminal:
-
- Install NLTK: run pip install -U nltk
- Test installation: run python then type import nltk
- Windows
-
- Install NLTK: http://pypi.python.org/pypi/nltk
- Test installation: Start>Python35, then type import nltk
- Download NLTK data
- import nltk
- nltk.download()
- from nltk.corpus import stopwords
-
- stopwords.words('english')[0:500:25]
-

- # NLP Basics: Reading in text data & why do we need to clean the text?
- # NLP Basics: Exploring the dataset
- # NLP Basics: Learning how to use regular expressions
- # NLP Basics: Implementing a pipeline to clean text
- Pre-processing text data
- Cleaning up the text data is necessary to highlight attributes that you're going to want your machine learning system to pick up on. Cleaning (or pre-processing) the data typically consists of a number of steps:
-
- Remove punctuation
- Tokenization
- Remove stopwords
- Lemmatize/Stem

- # Supplemental Data Cleaning: Using Stemming
- import nltk: ps = nltk.PorterStemmer()
  - print(ps.stem('grows'))
- # Supplemental Data Cleaning: Using a Lemmatizer
  - Test out WordNet lemmatizer (read more about WordNet here)
  - import nltk, wn = nltk.WordNetLemmatizer(), ps = nltk.PorterStemmer()

- print(ps.stem('meanness'))
- # Vectorizing Raw Data: Count Vectorization
- ### Count vectorization
- Creates a document-term matrix where the entry of each cell will be a count of the number of times that word occurred in that document.
- # Vectorizing Raw Data: N-Grams
- ### N-Grams
-
- Creates a document-term matrix where counts still occupy the cell but instead of the columns representing single terms, they represent all combinations of adjacent words of length n in your text.
-
- "NLP is an interesting topic"

- # Vectorizing Raw Data: TF-IDF
- TF-IDF
- Creates a document-term matrix where the columns represent single unique terms (unigrams) but the cell represents a weighting meant to represent how important a word is to a document
-

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i}$$

$tf_{i,j}$ = number of times i occurs in j divided by total number of terms in j
df_i = number of documents containing i
N = total number of documents

- # Feature Engineering: Feature Creation
- # Feature Engineering: Transformations
- Process
-
- Determine what range of exponents to test
- Apply each transformation to each value of your chosen feature
- Use some criteria to determine which of the transformations yield the best distribution

- # Building Machine Learning Classifiers: Building a basic Random Forest model
- # Building Machine Learning Classifiers: Random Forest on a holdout test set
- Explore RandomForestClassifier through Holdout Set
- from sklearn.metrics import precision_recall_fscore_support as score
- from sklearn.model_selection import train_test_split

- Building Machine Learning Classifiers: Explore Random Forest model with grid-search
- Grid-search: Exhaustively search all parameter combinations in a given grid to determine the best model
- Building Machine Learning Classifiers: Evaluate Random Forest with GridSearchCV
- Grid-search: Exhaustively search all parameter combinations in a given grid to determine the best model.
-
- Cross-validation: Divide a dataset into k subsets and repeat the holdout method k times where a different subset is used as the holdout set in each iteration.

- Building Machine Learning Classifiers: Explore Gradient Boosting model with grid-search
- Grid-search: Exhaustively search all parameter combinations in a given grid to determine the best model.
- Explore GradientBoostingClassifier Attributes & Hyperparameters
- Building Machine Learning Classifiers: Evaluate Gradient Boosting with GridSearchCV
- Grid-search: Exhaustively search all parameter combinations in a given grid to determine the best model.
-
- Cross-validation: Divide a dataset into k subsets and repeat the holdout method k times where a different subset is used as the holdout set in each iteration.

- Building Machine Learning Classifiers: Model selection
- Read in & clean text
- Building Machine Learning Classifiers: Model selection

Deep

- Word Encodings¶
- The notebooks explains the implementation of word encodings in NLP using the tensorflow library.
- Creating sequences of tokens
- The notebook covers the creation of sequences of tokens from words in a sentence.
-
- from tensorflow.keras.preprocessing.text import Tokenizer
- from tensorflow.keras.preprocessing.text import Tokenizer

- Padding the sequences
- The notebook explains how to manipulate sequences to make them of equal length using padding.
-
- Import the APIs
- ##import the required APIs
- from tensorflow.keras.preprocessing.text import Tokenizer
- from tensorflow.keras.preprocessing.sequence import pad_sequences

- Sentiment Analysis - Tokenizing news headlines for data preparation!¶
- The notebook covers the data preparation step by tokenizing the headlines and creating padded sequences of news headlines.
-
- Data preparation include the following steps:
-
- Download and read the data
- Segregate the headlines and their labels.
- Tokenize the headlines
- Create sequences and add padding.
- 1. Download and read the news headlines data
- This is a kaggle dataset which is further corrected and then hosted on Google Cloud Storage.

- Word Embeddings for Sentiment Analysis¶
- This notebook explains an introduction to word embeddings. We will train our own word embeddings using a simple Keras model for a sentiment classification task.
-
- Steps include:
-
- Downloading data from tensorflow dataset.
- Segregating training and testing sentences & labels.
- Data preparation to padded sequences
- Defining out Keras model with an Embedding layer.
- Train the model and explore the weights from the embedding layer.

- Projecting embeddings on TensorFlow projector
- This notebook explains how you can write the vectors of an embeddings into a TSV file to visualise it in a 3D space on the TensorFlow projector
-

- Classifying News Headlines¶
- This notebook explains the classification of news headlines as sarcastic and non-sarcastic. We are using the same headlines data as used before.
- -
- Text Classification challenge¶
- You are required to train a deep learning model on the IMDB reviews dataset and classify a set of new reviews as positive(1) or negative(0) using the trained model.

- Text Classification challenge
- You are required to train a deep learning model on the IMDB reviews dataset and classify a set of new reviews as positive(1) or negative(0) using the trained model.
- ---
- Implementing LSTMs using TensorFlow¶
- This notebook walks you through the implementation of an LSTM model to classify news headlines as sarcastic or not_sarcastic. We will analyse the accuracy & loss curves for training and validation sets.

- Improving the performance of the Text Classifier with CNN
- This notebook covers tries to explore the CNN model by replacing the LSTM model implemented in the previous video. We'll compare the accuracy and loss for a CNN model on the same headlines data.
- ---
- Yelp Review Classifier
- This notebook serves as a challenge to implement and explore LSTM and Convolution model over the new Yelp review dataset. You have to fill up all the blanks with the hyperparameters that helps you get the best accuracy and loss.

- Explore the LSTM & CNN model with the following layers:
- Embedding layer
- Try two bidirectional LSTM layers or a Conv1D layer or both.
- Dense layer with 24 nodes
- Output Dense layer with sigmoid activation

- Introduction to Text generation¶
- This notebook explains how we can split a given corpus of data into features and labels and then train a neural network to predict the next word in a sentence.
-
- Create a corpus - break the text down to list of sentences.
- Create a word_index(vocabulary) from the text.
- Tokenize the data and create n-gram sequence for each sequence of the corpus.
- Pad those sequences.
- Segregate features from the sequences by reserving the last element of the array as labels.

- Poetry generation challenge¶
- This notebook serves as a challenge on how to create poetry like Shakespeare by leveraging RNNs(LSTMs). We'll be using the Shakerpeare poetry as the training data and then use the trained network to predict the next words.

tensorflow-working-with-nlp---tensorflow-working-with-nlp-2439112-main

- NLP use cases
-
- Classifying whole sentences
- Classifying each word in a sentence
- Answering a question
- Text summarization
- Fill in the blanks
- Translating from one language to another

- Challenge: NLP model size¶
- How many parameters does the BERT base uncased model have? Use the get_model_size function below to help you.
- If you know the number of parameters for a model, how might you be able to determine how much memory is required when running a model inference?
- If you wanted to run a GPT-3 175 billion inference. How much RAM would your infrastructure require.
- This should take you between 5-10 minutes

Another Set of Problems for Assignments

NLP Problems (Assignments) to Solve

- Regular expression to extract data and information from text
- Tokenization using NLTK or similar
  - Word token
  - Sentence tokenization
  - Utilize regular expression
- Lemmatization using NLTK
- Stemming using NLTK or similar
- Remove stopwords from text
  - Then tokenize
  - Do lemmatization and stemming after stop word removal

NLP Problems (Assignments) to Solve

- NLP: Write code to remove punctuations from text.
- NLTK: Take stop words list from library. Add your own stop words to the list
  - Then remove stop words from a text
- Write N-Gram (Bi Gram, Trigram) code using frequence as the measure
- Write N-Gram (Bi Gram, Trigram) code using collocated words as the measure
  - What are collation words
  - Find a library method or a 3rd party implementation, use it as well
  - Compare your output with the library/3rd party one
  - Print top few N-grams

NLP Problems (Assignments) to Solve

- Find about NLTK methods/features such as
  - ngram_fd
  - ngram_fd.items()
  - dir(trigrams)
  - help(trigrams.ngram_fd)
  - nbest
  - TrigramAssocMeasures
    - raw_freq
    - help(TrigramAssocMeasures)

- NLTK features for
  - MLE (Maximum Likelihood Estimate) and Laplace smoothing
- Implement
  - trigrams based smoothing using Laplace and Kneser Ney algorthms
- Implement
  - Laplace smoothing
    - Bigram, trigram
  - Measure the preplexity
  - Measure prepexity as
    - a) Logbase2Prob= Sum-for-all-trigrams(log2(P(w3|w1,w2)) (b)Ent=(-1/tokens-in-test)x Logbase2Prob (c) Power(2, (ent))

NLP Problems (Assignments) to Solve

- Write a program to predict the next few words
  - Based on bi-gram and tri-gram
  - Based on a sample text
  - Use train and test approach
- Study this implementation
  - https://github.com/smilli/kneser-ney
  - Utilize utenberg corpus is required
-

NLP Problems (Assignments) to Solve

- Using Penn Treebank, do POS tagging to a text
  - http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
- Find out what are these
  - Treebank corpus and Brown corpus
  - Can you use these for the POS tagging task above
- Write short essay on
  - Chunk grammar
  - http://www.nltk.org/book/ch07.html.
  -

NLP Problems (Assignments) to Solve

- From some example text
  - extract three different chunks of your choice: e.g., co-ocurrences of adjectives and nouns, co-occurences of determiner, adjectives and nouns, extractions of all types of nouns, etc.
  - Learn on how to train a custom tagger at
    - http://www.nltk.org/book/ch05.htm
- Train an HMM model on the sentences of Brown corpus. Find out the accuracy of your trained HMM model on the sentences in test data

NLP Problems (Assignments) to Solve

- Use NLTK's tagger to predict the tags and determine accuracy of prediction.
- Read on
  - Maximum Entropy Classifier (MaxEnt)
  - Why MaxEnt is highly accurate
- Read on a decision tree classifier for POS tagging.
  - http://nlpforhackers.io/training-pos-tagger/

NLP Problems (Assignments) to Solve

- Classify text using
  - Naive Bayes
- Use movie data from here and classify the reviwes to be positive or negative. Use train and test
  - http://ai.stanford.edu/~amaas/data/sentiment/
  - load_files scikit-learn
  - CountVectorizer
- Modify the above implementation
  - Get rid of the words occuring in more than 1000 documents
- Redo the above implementation after modifying the text
  - Such as add Not when you see a negative word, and till the first punctuation

NLP Problems (Assignments) to Solve

- Use the sentiment lexicon below
  - http://sentiment.christopherpotts.net/lexicons.html
  - Create two features
    - Positive words count
    - Negative words count
- Filter out some words using POS tagging
  - Keep adjectives, verbs, and nouns
  - And then try the sentiment analysis
  - i.e. text classification

NLP Problems (Assignments) to Solve

- Try this example on Neural Network and text classification
  - https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/ (another beginner tutorial)
  - Try to improve the accuracy
- Train a Neural network with embedding
  - Use the IMDB database as above
  - For text classification
  - Measure the performance and compare the result with Naive Bayes

NLP Problems (Assignments) to Solve

- Repeat the text classification after replacing negative words wit
  - NOT
- Glove: https://nlp.stanford.edu/projects/glove/
- Pretrained word to dense vectors
- Use glove for a multi-label classification problem
- Use glove and build a binary classifier
  - Pick one of the toxicity columns as the class
  - Classification for Wikipedia comments

NLP Problems (Assignments) to Solve

- “Make a binary classifier for each class, and assign multiple labels (classes) to each test record. Evalute your accuracy for multiple classes. This is little more work but is more rewarding as a learning experience.”
  - Use Glove
  - And wikipedia comments

NLP Problems (Assignments) to Solve

- Try the name entiry recognition information and example
  - https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/
  - https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus/version/4#ner_dataset.csv
- Create a Bi-directional LSTM (RNN)
  - For Name Entity Recognition
- Modify the NER example using
  - Glove
- Concatenate each word with POS tagging
  - And then adjust the LSTM/RNN for NER

NLP Problems (Assignments) to Solve

- Implement Knee/Elbow method of text clustering
- Determine purity of clusters
- Utilize Bernoulli mixture model of clustering
- Explain the Bernoulli mixture model of clustering model as can be seen in
  - https://github.com/manfredzab/bernoulli-mixture-models
  - https://github.com/schwannden/MNIST_mixture-of-bernoulli

NLP Problems (Assignments) to Solve

- Gaussian Mixture models of clustering
  - http://scikit-learn.org/stable/modules/mixture.html#mixture
- Implement LDA topic modeling algorithm
  - Utilize train/test
- Implement PLSA topic modeling algorithm
- Utilize PLSA topic modeling algorithm from Scikit-learn and apply on a dataset
-

NLP Problems (Assignments) to Solve

- Read the Topic Modeling blog at
  - https://nlpforhackers.io/topic-modeling/
- Implement
  - CountVectorizer (Freq)
- Take a set of text/articles/news
  - Train LDA on the data and find the top topics
  - Apply EM if applicable
- Implement PLSA topic modeling algorithm
  - TruncatedSVD
- Modify the code in the URL to get rid of noise from the tokens. Remove *,/,-,=,,_ or similar

NLP Problems (Assignments) to Solve

- Implement LDA and PLSA
  - And apply on some Gutenberg project data
  - Find the topics
- Use LDA, PLSA to find topics
  - Use these topics as features for Naive Bayes
  - Then implement a Naive Bayes classifier
  - Find: accuracy, precision and recall

NLP Problems (Assignments) to Solve

- Read on Gensim
  - https://radimrehurek.com/gensim/models/ldamodel.html
  - Implement LDA
  - Use alpha and beta
  - Use Gensim

NLP Problems (Assignments) to Solve

- Implement Textrank algorithm
  - Use Gensim
- Take articles from the Internet
  - Implement text summarization
  - Implement keyword extraction using Gensim
- Implement Naive Bayes classifier as below
  - Find 100 key phrases/keywords from each document/review
  - Merge these keywords and
  - Filter these keywords from the documents
  - Then with the remaining data train and implement Naive Bayes
    - Calculate accuracy, precision, and recall
- Implement ROUGE metric for text summarization
  - “ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing” Wikipedia