**Advanced Natural Language Processing Skills Training**

**Jan 25-27, 2021**

**Paul Rodrigues, PhD**

**Applied Research Group**

**Accenture Federal Services**

In general, Day 1 focuses on what makes NLP different than other areas of data science. Day 2 focuses on statistical NLP. Day 3 focuses on deep neural embeddings.

**Day 1**

Overview

Presentation: Linguistics for NLP

Case Study: Human Resources NLP

LUNCH

Case Study Discussion

Presentation: Overview of NLP COI Software, Learning, and Dataset Resources [NO SLIDES]

Case Study: Linguistic Features

Presentation: Notebook Services for Learning NLP

Demonstration of Google Colab [NO SLIDES]

Programming Challenge: Get familiar with NLP in Google Colab

**Day 2**

Presentation: Unicode – What you need to know

Presentation: Unicode Supplement

Programming Challenge: Build a Language ID System

LUNCH

Programming Challenge: Build (Extend) a Language ID System

Presentation: Differences between NLP and Text Analytics (Liz)

Presentation: Black Box NLP

?? Intro to Probability Theory / Chain Rule

?? Sentiment

Presentation: Advanced Topics in Topic Modeling

Programming Challenge: Topic Modeling

LUNCH

Discussion: Propose a new Emoji Character

**Day 3**

Presentation: How to get to Sesame Street

?? Independent Study: Beginner and Advanced Track BERT Videos

LUNCH

Programming Challenge: Sentiment Classifier using BERT

Presentation: NLP Visualization

Discussion: Projects at AFS using NLP

1. Introductions
2. Questions
   1. When combining BERT with other data, is the best way to extract the BERT embeddings and then combine with the other data in vector form and run through your DNN, or is there a more efficient way to use that other data in BertForSequenceClassification?
      1. LinkedIn created Dtext to utilize external features in addition to text on the page
   2. Any good way for dealing with imbalanced labels with BERT? I’ve heard DICE, but have not seen any good real world results.
      1. BERT is binary cross entropy at default – dice loss?
      2. There has not been a lot
      3. Can do data augmentation: https://github.com/makcedward/nlpaug
   3. Which embeddings best to pull out from a model run on multiple epochs? Where training and valid loss are closest?
      1. Yes, basically. Just what I did.
   4. Recommendations for getting explanations?
      1. Same as I am already thinking
   5. What about the embeddings for valid and test? Does it make sense to pass those into another architecture with other data like account data?
      1. So you need to pass it through the train embedding, which is what I did I think
   6. How to integrate a time series component?
3. Intro Session – 1/25/21 AM
   1. Mutual intelligibility is the standard for differentiating languages – when one is spoken can it be understood by the other – A Portuguese speaker can understand a Spanish speaker but not visa versa usually
   2. Resolution – structured matching (kind of utilizes NER), database matching, fuzzy matching, geo matching, etc.
   3. The link across all the NLP applications is structure
   4. Linguistics background
      1. Phonetics – study of the sounds of human language
         1. NLP involving sound (might relate to audio things like Alexa maybe)
         2. Bilabial at the front all the way to glottal at the back
         3. Arabic can’t really have lots of consonant clusters – str not really a thing for example
      2. Orthography – how people document their language
         1. Text, spelling
         2. With English, we use an alphabet
         3. Others might use a syllabary (a sound that represents a syllable) – Japanese katakana
         4. Symbols (Chinese)
      3. Semantics
         1. Something can be syntactically valid but not semantically valid
            1. Angry red crabs snap furiously – valid
            2. Colorless green ideas sleep furiously – syntactically valid but not semantically valid
      4. Higher level – means deeper down the linguistic chain
4. Breakout rooms – 1/25/21 PM
   1. Org is having an HR issue and they are trying to recruit participants
      1. Brainstorm several different levels of proposal using NLP, ML, DatSci
      2. Special employees
      3. Analytical employees
   2. NLP applications to the case study of candidate recruitment and hiring for FBI
      1. Social media
      2. Semantic similarity
      3. If people are actively participating in competitive sports, it shows that an individual is more likely to go through and complete all the phases of the project
   3. These are people that they are monitoring for four years – where should they delegate the limited resources?
      1. This client would spend more on solutions that look forward and help reduce the burden so they know who to focus on
   4. Using a network analysis approach maybe – if no worries about privacy
      1. If I get a clearance, someone else I know can. Who I went to school with also similar skillset.
         1. Can create bias already
   5. Can you target shows popular to a specific population for diversity or age based recruitment?
   6. Voice recognition elevator video
      1. The strength of a model is based on what it is trained on
         1. Keep saying it until it understand Scottish – active learning
      2. Mozilla project trying to elicit voice samples from around the world – lots of organized attempts at solving this in the future
         1. On the text side there is less organization, but more data
      3. Develop approaches to actively search out what you don’t have
      4. You can use surveys and identify people before hand
   7. There is a part of NLP that identifies age, gender, etc. from language – meaning that his exists at human level too (think the resume problem)
   8. NLP resources in the COI
      1. Cloud providers are putting out something on form OCR
      2. Natural language generation – the ability to spit out text that is about a particular subject
         1. GPT-2 is good at generating long text and analogies (Out of OpenAI)
            1. GPT-3 is latest and greatest
      3. Grover – detecting generated fake news
         1. You might be able to identify common factors amongst news that were generated from the same generators (like GPT-2)
            1. Coherence ends up being a pretty good track
            2. Easier to catch coherence on longer texts than on Tweets
      4. Rasa intent recognition
      5. WebAnno also really good for labeling and annotation – Like Prodigy
      6. Appen is like Premise in that people can go to a remote local village and record data – it’s a linguists dataset creation forum
         1. Paul is really good to help us with custom dataset need (training sets)
      7. Paul gets pretty concerned when we anyone wants to use prepacked pre-canned
5. Breakout two – 1/25/20 PM
   1. Identify who it was that leaked the material – author identification
      1. You’ll need some type of data where the author is identified and then you can match the style of that to the anonymous writing
         1. BERT is one example
   2. Author profiling
      1. You’ve got metadata, a social graph, etc.
   3. Linguistic resources
      1. Any features that are specific to English that would require a database to look up information or other classifiers to help your classifier
      2. Syntactic feature would require English model maybe?
         1. So you can get at these features statistically or via knowledge of the language
   4. Tomorrow is statistical NLP and programming on Google Colab
6. Day 2 – 1/26/21 – AM
   1. Intro to Probability Theory
      1. Random process – e.g. shuffles, coin tosses, etc.
      2. Can be helpful to model a problem of random even if not random
         1. P(A) = Probability of an event
      3. Frequentist interpretation – probability of an outcome is the proportion of times the outcome would occur if we observed the random process an infinite number of times
      4. Bayesian interpretation – Interprets probability as a subjective degree of belief. Allows for prior information to be integrated into the information framework
      5. Might use frequentist when you have a set corpus and set test (you know that there is a finite number) but bayesian when you had unrepresented findings in your test data
         1. Supervised might be more frequentist and unsupervised more Bayesian (this is general)
            1. NER, parsing frequentist, but topic modelig and active learning frequently Bayesian
      6. Law of large numbers – as more observations are collected, the proportion of occurrences with a particular outcome converges to the probability of the outcome
      7. Common misunderstanding – gambler’s fallacy/law of averages
         1. The idea is that the future will compensate for the past (I’m due for a win given all the bad in the past)
   2. Intro to n-grams
      1. Language modeling – assign a probability to a sentence
         1. So like The office is fifteen minutes from my house is more likely than the office is fifteen minuets from my house
         2. Given a sequence of words, compute the probability -> P(W) = P(w1,w2,w3,w4,w5…wn)
         3. Can also do probability of the next word -> P(w5|w1,w2,w3,w4)
      2. Relies on chain rule of probability
         1. P(A|B) = P(A,B)/P(B) or P(A|B) \* P(B) = P(A,B) or P(A,B) = P(A|B)P(B)
         2. The chain rule can be applied to compute joint probability of words in sentences
            1. P(“it’s water is so transparent”)
      3. How to estimate the probabilities?
         1. P(the|it’s wated is so transparent that)
         2. Can’t just count all because too much combinatorial
         3. We use Markhov assumption – We estimate the probability of (the|it’s water is so transparent that) just be computing P(the|that) or P(the|transparent that)
            1. Look at couple previous words rather than entire
            2. We approximate each component of the product
      4. Simplest case: Unigram model
         1. By product of probability of individual words (unigram)
      5. Bigram model
         1. We condition on the single previous word
      6. N-gram model
         1. Language has long-distance dependencies, so in general this is insufficient
         2. In practice, trigrams and maybe a bit above are just constraining enough
         3. Which N is a better approximation of a document?
            1. The longer the better, but reducing may better fit entire corpus
   3. Estimating N-gram Probabilities
      1. MLE = count(wi-1, wi)/count(wi-1)
         1. Example
            1. P(I|<s>): 2/3 – I starts the sentence in 2 out of 3 sentences
            2. P(Same|am): How many times does am, Sam occur? 1 am, Sam occurrence over 2 am occurrences)
         2. Bigram count table – I want is a common occurrence vs. something like food want is 0 (food never followed by the words want)
            1. Normalize by unigram count to turn into probabilities
         3. Why is English|want lower than Chinese|want -> Because people want Chinese food a lot (about the world)
         4. P(to|want) is .66 and P(want|spend) is 0 -> This one is about grammar
         5. P(food|to): This is grammatically possible but not in our corpus (contingent)
      2. In practice, we use log probabilities to avoid underflow (0s)
      3. Language modeling toolkits: SRILM, Google N-gram release, Google book n-gram corpus
   4. Evaluation and Perplexity
      1. What does it mean for a language model to be a good language model?
         1. Assign higher probability to real or frequently observed vs. unreal or rarely observed
         2. We train parameters of a language model on a training set and test on unseen data
         3. We need an evaluation metric then
      2. Best is to put models in a task – speech recognizer, MT system, spelling corrector
      3. Metrics: How many misspelled words corrected properly, translated correctly, etc.
      4. Extrinsic evaluation (in-vivo)
         1. Problem is that it is time consuming
      5. Instead, we’ll use an intrinsic evaluation
         1. Perplexity – Bad approximation to extrinsic unless test data looks a lot like train
            1. Generally useful only in pilot experiments
            2. Good when we use extrinsic as well
            3. Best language model is one that best predicts an unseen test set

I knew that was coming

* + - * 1. Perplexity – probability of test set normalized by number of words
        2. Perplexity of a string of words is the nth root of the product of n bigram probabilities multiplied together and inverted (a function of the probability of a sentence)
        3. Minimizing perplexity is the same as maximizing probability
      1. Perplexity is related to the average branching factor
         1. On average, how many things can occur next?
         2. High perplexity when it is hard to guess the next digit or name out of 30,000 names in a sequence for example
      2. Perplexity is weighted equivalent branching factor
      3. Lower perplexity equals better model
         1. Unigram will have higher perplexity, bigram lower, and trigram even lower

1. Unicode
   1. UTF-8 used most unless you are working on an Asian language like Chinese, where UTF-6 is better
      1. UTF-8 takes less space on your hard drive
      2. Asian logographic characters are usually multibyte
   2. Circle of dots means a character is a diacritic
   3. There are at least 8 different versions of Unicode with Arabic (Actually 13)
   4. Unicode consortium manages the emoji too
   5. Emoji processing
      1. Can utilize categories
      2. Catalogues categorize emotional categories of emojis – but might be cross culture problems
2. Afternoon breakouts – 1/26/21
   1. Build a language detector – we have to build a language id system
   2. We are doing 3 day language
   3. Visualization Tools
      1. Lime – Can be used to highlight the terms or evidence that help a classifier classify a document (going down the tree on LDA)
      2. Scattertext can diagram against multiple dimensions of data and highlight categorical information
   4. Eli5
      1. Explaining modules with their own algorithms – different than the stuff I’m looking at for
   5. Bertviz
3. Sentiment Analysis
   1. VADER is a lexicon based approach for VADER
4. Day 3 – Deep Neural Networks
   1. Video
      1. Multilayer perceptron is most simple at front
      2. Definitions
         1. Neurons – thing that holds a number
            1. 28 x 28 pixels = 784 neurons holding a number representing the greyscale value of the corresponding pixel

The number inside the neuron is called its activation

Activation basically lit up when pixel is high

* + - * 1. This is our first layer
      1. Last layer is a number of neruons based on output classes
      2. Hidden layers in between
         1. This example uses 2 layers each with 16 neurons
         2. Activation in one layer determines activation in the next layer – there is a specific how

Have some mechanism that could combine patterns into edges into digits

What parameters should the network have?

Assign a weight to the edges between the neurons in layers next to each other

Then get the computed sum from these weights

So basically adding up the values of the pixels in the region we care about (and then negative ones around that region to help differentiate)

Then pump this weighted sum into a function that squishes it into a number between 0 and 1 -> Sigmoid or logistic curve can do this

You might want bias for inactivity

Add in another number to the weighted sum before plugging it into the sigmoid function – the bias term (how high the weighted sum needs to be before the neuron starts getting meaningfully active)

* + - * 1. So in our example, we end up with 13,002
        2. Learning -> Finding the right biases and weights
    1. Why the layers?
       1. We’re piecing together various components (like a loop with a 9 and then a line below)
          1. Then in the layer before that it’s the little edges in that loop or line, and so on
       2. So a specific neuron activation will be close to 1, making going from that layer to last one just based on which correspondent of shapes corresponds to digits
    2. It is more accurate to treat each neuron as a function – takes the inputs and spits out a number between 0 or 1 for example
    3. Sigmoid function
       1. ReLu seems to be (rectified linear unit)
          1. Take a max of 0 and a where a is given by the network
          2. Motivated by a biological analogy by how neurons would be activated or not
          3. If passes a certain threshold, it would be the identity function
          4. Sigmoids made it very difficult to train at some point with deep networks
    4. What is an NLP deep learning model capturing?
       1. Features that were traditionally used in NLP processing pipelines
       2. How many layers in DNN for NLP?
          1. Some of the most common are 12, but it’s a thing you have to try for your problem, experiment with to determine how many you need
          2. No straight forward answer
       3. Method setting the weight can be adjusted, bias can be adjusted manually, depth of the network
  1. Presentation
     1. 2013 changed everything – Word2Vec
        1. Instead of using a bag of words, you use a vector space – now you can do mathematics on words
        2. Queen – women + man = King
        3. Paris to France as is London to England
        4. So now every words has a number – location in vector space, word similarity
           1. N-grams wouldn’t capture this
     2. But 2018 REALLY changed everything by leveraging tricks that were developed for computer vision
        1. Muppets arrived and started solving everything
        2. CV introduced stack or deep learn and more importantly, transfer learning
        3. You can create this massive NN and then distribute it and tweak it (fine tuning)
           1. Adjust the weights and the final layer
           2. You can take a NN created off of a database and then add another layer that predicts your particular problem
     3. NLP is in a bit of a renaissance right now, so it’s hard to keep up
        1. GLUE benchmarks – new one every week
           1. BERT came out and annihilated the benchmarks, so we had to create superglue
        2. You need intensive compute power to do most things now – weeks and millions of dollars
        3. Will be too expensive to create our own models with clients
        4. So only needed if you are looking at completely new unique data that pre-trained won’t be useful for
     4. ULMFit (2018) and then ELMO (2018) and then BERT (2018) and then ERNIE (2019)
        1. Two layer bidirectional LSTM innovation with ELMO
        2. BERT used transformers, which were lighter weight
           1. Bigger innovation was the joint conditioning of the left and right context
           2. Trained system by blanking out word and having system guess what word was used to fill in the blank
           3. Read from right to left and then left to right at same time as opposed to chronologically
        3. Enhanced Representation through kNowledge IntEgration (ERNIE)
           1. Not used often here as it is Chinese software and it is complex
           2. Continuous training is a big innovation

Another innovation is that is could do multiple tasks in the same model

* + - * 1. This one is ERNIEs in this space – Baidu’s is PaddlePaddle/ERNIE
      1. RoBERTa from Facebook, who managed PyTorch
         1. It’s really just BERT with more training data, larger learning rate, removed next-sentence prediction
         2. Higher accuracy for English
         3. So if you are using English, you can just use this
      2. Previous CTO liked Tensorflow more than PyTorch
         1. Tensorflow more popular in CV and Pytorch more popular in NLP
      3. DeepPavlov is Russian NLP
      4. HuggingFace Tranformers
         1. Select which model you want to use on which platform you want to use
    1. Custom hardware companies are constantly just acquired by NVIDIA
       1. Cerebras.net
  1. Classification with BERT
     1. Cased vs. uncased
        1. Uncased typically works best unless case is an important signal – author profiling for example
     2. You don’t have to take text and do a lot of preprocessing of the text
        1. Wordpiece – the go to segmentation method: Strings that are statistically important at the time of training
        2. Sentencepiece – What BERT was supposed to do originally
        3. Byte Pair Encoding
        4. Unigram Language model
     3. Substrings used in the model can be found in the vocab file
        1. Don’t use a multilingual model if a monolingual model is available
        2. So that every single piece in vocab files are relevant to your question
     4. Use Roberts for English as the accuracy will be higher
        1. Accuracy tradeoff
        2. LabSe prints out embeddings
     5. In the future, sbert might be better for embeddings extraction
     6. BertLangStreet and Adapterhub are good resources as well
     7. Bert Tutorial – The one that I’ve done
     8. After running this step, last\_hidden\_states holds the outputs of DistilBERT. It is a tuple with the shape (number of examples, max number of tokens in the sequence, number of hidden units in the DistilBERT model). In our case, this will be 2000 (since we only limited ourselves to 2000 examples), 66 (which is the number of tokens in the longest sequence from the 2000 examples), 768 (the number of hidden units in the DistilBERT model).
        1. For sentence classification, we’re only only interested in BERT’s output for the [CLS] token, so we select that slice of the cube and discard everything else.
     9. What are the 768 embeddings?
        1. There are 768 dimensions of the embeddings, which represent each of the neurons in that layer
        2. So you’re taking the meaning behind the sentence instead of just the word alone
        3. The CLS token is the target output of the classifier
  2. USDA example
     1. They used a chron job for the Python component

1. BERT Fine-tuning
   1. Attention (attention masks) – Pay attention to this over here