Class 401: Shanqing Gu
Submission date: 07/08/2019

SMU MSDS 7337 Mid-Term Exam Summer 2019

I. Short Essay Responses

(25 pts each, 300-500 words each).

1. Select one career or industry that makes use of applied NLP.
    a. Explain generally how that field or career utilizes NLP.
    b. Explain at least some methods of NLP that are very likely to be used in the career or industry you selected.
    c. Give at least one specific example of a use case for NLP within the chosen field, and explain how the problem or situation is (or could be) improved by applying NLP.

Question I.1 Response: (471 words)

In November 2018, Amazon launched Comprehend Medical service (ACM). This ACM service uses pre-trained natural language processing (NLP) models to understand the medical information and identify meaningful relationships in unstructured text. It provides some different real-time APIs for key phrase extraction, sentiment analysis, entity recognition, topic modeling and language detection. The application output is exported in JSON format. There are 3 major advantages for ACM service: (1) extracting medical information quickly and accurately, (2) protecting patient information, and (3) lowering medical document processing costs.

The ACM service applies varied NLP methods for different APIs. For examples: (1) the Syntax analysis API is used to analyze text using tokenization and Parts of Speech (POS). Tokenization is to identify word boundaries, and POS to make labels like nouns and adjectives within the text. (2) the Entity Recognition API uses chunks to segment multi-token sequence and automatically categorize the name entities (“People”, “Places”, “Locations”, etc.) appropriately, and (3) medical named entity and relationship extraction (NERe) API returns the medical information such as medication, test and treatment. This API also identifies relationships between extracted subtypes associated to medication.

Importantly, the Custom Classification API can enable customers to easily build text classification models in a two-step process. The first step is to train the classifiers of interest with up to 1,000 training documents for each label. Thee testing results will return in metrics to improve models with accuracy, precision, recall and macro F1 score. Deficiencies in precision or recall can be further addressed by expanding the training documents. The second step is to deploy the most optimized machine learning models with submitted documents for classification.

Early ACM customers like Fred Hutchison Cancer Research Center (FHCRC) and LexisNexis used the Custom Classification API for their specific purposes. In FHCRC, this service was used to match patients for specific cancer clinical trials from millions of clinical notes. It only takes one hour to process 11,000 documents with same accuracy as expert. For LexisNexis, this service is used to build custom entity recognition models which can identify quickly from more than 200 million documents at greater than 92 percent accuracy.

More customers are working with ACM in in different NLP fields now. Here just name a few on the list: Vibes, Roche, FINRA, PwC, VidMob, Deloitte, PubNub, ClearDATA, and TINT. For examples: (1) Vibes uses ACM to quickly extract key phrases, detect sentiment, and model topics from unstructured message comments; (2) PwC uses ACM to build smarter application and extract critical insights with less annotating, and (3) Roche uses ACM to quickly extract and structure information from medical documents.

With applying different NLP models in ACM services, all these customers can retrieve valuable information which is otherwise difficult without using significant manual effort.

References:

1. Amazon comprehend medical: https://aws.amazon.com/comprehend/medical/
2. Amazon comprehend features: https://aws.amazon.com/comprehend/features/
3. Amazon comprehend documentation: https://docs.aws.amazon.com/comprehend/latest/dg/comprehend-dg.pdf

2. Choose one of the "trade-offs" in NLP that was covered in the materials for this course.
    a. Explain the trade-off in general terms. Define the two choices.
    b. Explain the benefits and weaknesses of each side of the trade-off. Include at least one benefit and one weakness of each.
    c. Describe a work-situation that would make one of the choices in the trade-off much better, in terms of practical outcomes for you and your stakeholders on a project.

Question I.2 Response: (462 words)

How to balance the trade-off between feature learning (a.k.a. representation learning) and feature engineering (a.k.a. feature extraction) in Natural Language Processing (NLP) is the actual problem to consider when starting a project. Both of them share some overlaps in data science workflows, but definitely distinct in their objectives.

Feature learning is an automatic identification and use of features from raw data to representation. It can either supervised or unsupervised in feature learning. The concept of feature learning was successfully used in the convolutional neural networks for classification and in autoencoder for learning generative models of data. In NLP, features can be learnt from annotated documents rather than being specified and extracted by subject matter experts (SMEs). Feature learning is very convenient and advantageous to reuse features especially for multi-task learning when a training dataset already exists (or can be collected passively). The disadvantages are the features are usually simple and a large amount of data are required for feature learning process.

Comparatively, feature engineering is a process to extract and select useful features from raw data for creating machine learning models in better model approximation. It requires highly competent SMEs with domain knowledge of the documents to classify. Sometimes, just one excellent feature could be the clue for better classification. For unstructured and textual data, different strategies can be used from models like: Bag of Words model, Bag of N-Grams mode, TF-IDF model, similarity features, topic models, Word2Vec, GloVe, and FastText. These tools can help convert free-flowing text into numeric representations for building more complex models. The disadvantages are handcrafting features is time-consuming and usually difficult. In addition, symbolic representation (grammar rules) makes NLP system fragile.

In the application of neural machine translation (NMT) deep learning-based approaches, machine translation can learn sequence-to-sequence transformations directly. A recurrent neural network (RNN) is typically used for the word sequence modeling. A bidirectional RNN (encoder) is used to encode a source sentence for a second RNN (decoder). The decoder is used to predict words in the target language. For long continuous sentence translation, it is done with convolutional neural networks with "attention" based approaches. Basically, the process pipeline of NMT eliminates hyperparameters and manual feature engineering. This example shows feature learning perform better than feature engineering in NMT approaches.

In addition, the trade-off for using either feature learning or feature engineering in natural language processing depends on the datasets and the features to use. The algorithms do not always work well with too many features. Therefore, the ideal approach is to implement the best of both approaches together and initiate by bootstrapping the feature-engineering process with candidate feature learning.

References:

1. Chen et al., The best of both worlds: combining recent advances in neural machine translation. https://aclweb.org/anthology/P18-1008
2. SMU Natural language processing Unit 3 lecture notes.

II. Essay Response

(50 pts, 600-1000 word essay).

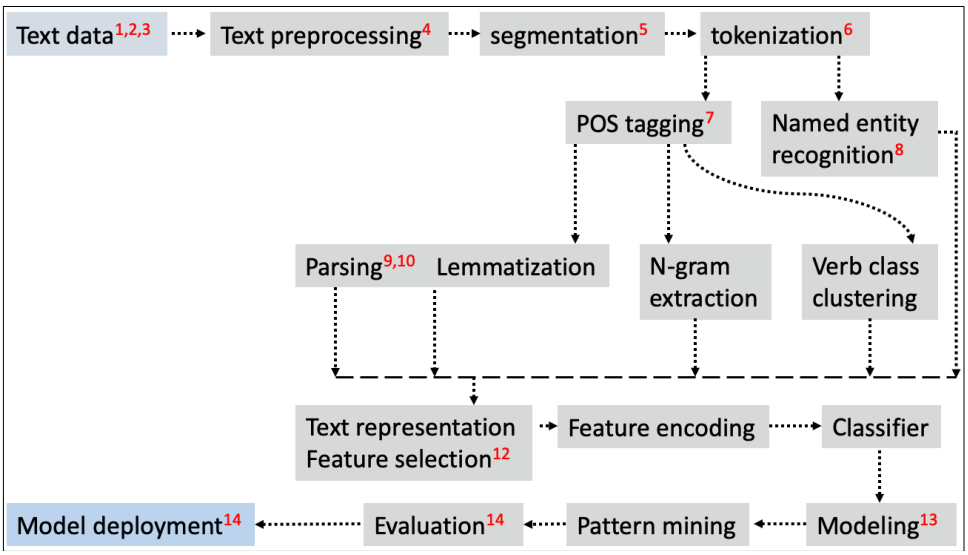
1. Describe the overall NLP "pipeline" for a text analytics project as described in the materials of this course. First give an overview, then briefly describe and explain each phase of the pipeline.
2. For each phase of the pipeline, either (a) recommend a specific tool for getting the job done, e.g. a Python package, and a reason why you recommend it, or (b) explain a choice that must be made, where you would configure the process for that step very differently depending on the kind of application.

Note: For 2, do each of (a) and (b) at least once.

Question II Response: (977 words)

Text analysis (a.k.a. text mining) is the automated process of extracting and classifying information from unstructured text data, such as online reviews and survey responses. For efficient text mining, the major steps of overall NLP pipeline (e.g. text classification) are

illustrated in the flowchart (below) and further discussed in detail with each pipeline component. Each component is listed accordingly to the superscripted number in flowchart.



1. Problem statement:
   1.1. Use domain knowledge to identify appropriately the specific issues to be addressed or a condition to be improved. This will help the project team to understand the problem and work together toward developing a solution.

2. Setting up dependencies in Python environments:
   Import major Python libraries for NLP: Requests, BeautifulSoup, pandas, numpy, matplotlib.pyplot, seaborn, spaCy, NLTK, Pattern, Stanford, etc..

3. Text data retrieval:
   3.1. Source of text data can either be internal or external. For internet data, need to export it to a CSV fie or retrieve the data with an API. For external data, need to use web scarping tools to open websites and collect data from different URLs.
   3.2. BeautifulSoup library is used to parse HTML and XML data. 3 features make BeautifulSoup powerful: (a) provides a few simple methods and idioms for navigating, searching and modifying a parse tree, (b) automatically converts incoming documents to Unicode and outgoing documents to UTF-8, (c) sits on the top of popular Python parsers like lxml and html5lib which allow to use different strategies or trade speed for flexibility.
   3.3. The data are further cleaned by removing the markup and select a slice of characters.

4. Text preprocessing and normalization:
   4.1. Remove unnecessary content from one or more text documents in the corpus (or corpora) and get clean text documents.
   4.2. Build a text normalizer which can remove HTML tags, remove accented characters, expand contractions, remove special characters, make case conversions, make text

correction, correct repeating characters, correct spellings, stem and lemmatize, and remove stopwords.

5. Sentence segmentation:
   5.1. Divide text into sentences.
   5.2. Identify sentence boundaries between word in different sentences (use spaCy).
   5.3. Useful for tasks: sentiment analysis, question answering systems, relation extraction.

6. Tokenization:
   6.1. The process of breaking up a string of characters into semantically meaningful parts that can be analyzed while discarding meaningless chunks.
   6.2. NLTK recommends the default sentence tokenization function nltk.sent_tokenize. NLTK and spaCy can be used together using some of their robust utilities to build a custom function to perform both sentence and word-level tokenization.

7. POS tagging:
   7.1. The process of assigning each detected token with a grammatical category, such as adjective, noun, verb. It is important for syntactic and semantic analysis.
   7.2. Different tools can be selected from NLTK, spaCy, Pattern, Stanford.
   7.3. NLTK is a great toolkit for teaching and learning NLP, Unigram POS tagger is a lookup tagger, it has been trained before being used to tag untagged corpora.
   7.4. spaCy focuses more on efficiency and performance for production use.
   7.5. Pattern is a web mining module for the Python programming language.
   7.6. Stanford POS Tagger is a Java-based implementation with a log-linear POS tagger.

8. Named entity recognition (NER): often performed using chunks which segment multitoken sequences, and label them with the appropriate entity type. Common types include organization, person, location, date, time, and GPE (geo-political entity).

9. Shallow parsing (a.k.a chunking):
   9.1. Chunking is to obtain semantically meaningful phrases and observe relations among them. There are 5 major categories of phrases (NP, VP, ADJP, ADVP, and PP).
   9.2. Chunks can be built with treebank corpus in NLTK, or parsetree function from pattern library. spaCy has Universal Dependencies Scheme and the CLEAR Style Dependency Scheme, and Stanford Parser generally uses a PCFG parser.

10. Constituency parsing and dependency parsing
    10.1. Constituent-based grammars are used to analyze and determine the constituents of a sentence.
    10.2. Dependency-based grammars to analyze and infer both structure and semantic dependencies and relationships between tokens in a sentence.

11. Exploratory data analysis: the approach to analyzing datasets to summarize the main characteristics, often with visual methods like box plots, scatter plots, etc.

12. Text representation and feature engineering:
    12.1. As shown in 1 and 2, understand text data, build a text corpus, and preprocessing the text corpus.
    12.2. Traditional feature engineering models: CBOW model, Bag of N-Gram model, TF-IDF model, similarity features, topic models. Traditional models have some limitations such as: feature explosion.
    12.3. Advanced models: word2vec, CBOW model, Skip-Gram model, GloVe, FastText.
    12.4. Advanced supervised deep learning models: CNN for sentence classification, document triage, and sentiment analysis, recurrent neural networks (RNNs), long short-term memory networks (LSTM), Gated recurrent unit (GRU), attention schema and transformer.
    12.5. For selecting deep learning models: (a) RNNs are a family of networks which are suitable for learning representations of sequential data like text in NLP, (b) LSTM networks address the problem of long-term dependency or remembering relevant information of the present output, (c) GRU is the simplified version of LSTM by combining the cell gate and hidden gate together, (d) Attention only uses parts of an input where the most relevant information is concentrated instead of an entire sentence.

13. Modeling and/or pattern mining:
    13.1. Select models based on problem statement, text data and features to use.
    13.2. Find optimal hyper-parameters suing grid search.
    13.3. Visualize models (i.e. pyLDAvis for topic models in an interactive visualization).

14. Evaluation and deployment (i.e. text classification)
    14.1. For text classification, one of the best ways to evaluate model performance is to visualize the model predictions in the form of a confusion matrix.
    14.2. For text classification, use Scikit-Learn to get the model's classification report.
    14.3. Deploy the model for prediction.

References:

1. Notes from Dr. Robert Slater's NLP classes and homework assignments
2. Dipanjan Sarkar. Text analytics with python. A practitioner's guide to natural language processing. The 2nd edition. Apress.
3. Rowel Atienza. Advanced deep learning with Keras. Packt.