Capstone Proposal

Carson Dahlberg August 9th, 2018

1 Domain Background – Natural Language Processing

Alan Turing published his famous article "Computing Machine Intelligence", in 1950, formalizing criteria to evaluate intelligence. A successful Turing Test included a machine performing so well in real-time written conversation, with a human, that said human would be unable to reliably distinguish between the program and a human. He firmly believed that machines could learn and achieve this level of intelligence.

My personal motivation for choosing this domain is that intelligent applications' potential is very intriguing, and the constant innovation is exciting. I want to lead the charge, where I work, to unleash these apps on our unstructured textual data, solving real-world problems and creating value by reduce costs, eliminate boring tasks, and aid/empower expert humans.

2 Problem Statement

Explore using machine learning models for solving the NLP-related tasks related to sentiment analysis (classifying polarity). The scope of this project will focus on practical models which must be explainable.

Compare the strengths, weaknesses, and trade offs when considering the models. I intend to create a framework for solving sentiment problems in the real world – including being able to communicate model's limitations, building an intuition, and explaining trade offs to inform the business' decision making.

- **Problem Definition**: taking social media reviews (movies, restaurants, products, and services) and
    1. categorize the polarity of text as positive/negative sentiment
    2. and predict the probability of classes
- **Quantifiable Tasks**: (Evaluation Metrics described in section 6) - the models should be able to predict the polarity of the text sentiment (positive versus negative).

3 Datasets and Input

- Respecting the reviewer's time to evaluate the project, I will only work on a subset of data, in order to keep the project size to a sensible size.
- **Data Guidelines in General** – the dataset will be balanced between negative and positive sentiment. The use of standardized dataset should help with this. Data Pipelines will be created to maintain consistent/repeatable data being sent to the models between training and testing.
- **Sentiment Labelled Sentences Dataset** – from UCI Machine Learning Repository.
    - This is the dataset used to train and test the models.
    - This was created for the Paper 'From Group to Individual Labels using Deep Features', Kotzias et. al,. KDD 2015. It contains sentences labelled with positive or negative sentiment, with scores of either 1 (for positive) or 0 (for negative). Sentences extracted from reviews of products, movies, and restaurants from three different websites/fields: **imdb.com**, **amazon.com**, and **yelp.com**. For each website, there are 500 positive and 500 negative sentences, selected randomly from their respective larger datasets, having a clearly positive or negative connotation. The goal was for no neutral sentences to be selected.
- I will select and label a small sample size of sentences from **UrbanDictionary.com** to examine model efficacy.

4 Solution Statement

Apply a survey of machine learning algorithms, contrasting performance between them in order to understand the trade offs involved in final model selection. Performance will be evaluated on predicting polarity of the sentence sentiment, summarizing strengths and weaknesses (time and data to train, for example). Traditional machine learning models will be used to derive the relationships between feature representations of text to predict/classify sentiment polarity and predicted class probability.

This problem will be solved through multiple steps. The first step is to build a preprocessing pipeline for the data into various formats which the models might require and/or use. This will also allow for identifying if specific feature engineering improves upon a model's metrics. Second will be training/validation/testing various models for the task, and then optimizing the models. Finally, results will be aggregated and summarized.

5 Benchmark Model

- Naïve model – consistently guessing/predicting same (majority) class
- Simple Models – default models without hyperparameter tuning

6 Evaluation Metrics

- **Overall Accuracy** – the percentage of text classified correctly (positive/negative sentiment).
- **Class-Level Precision** – the percentage of all text classified as belonging to a given class that actually belong to that class.
- **Class-Level Recall** – the percentage of text associated with a particular class that were classified as such. I will examine both Positive and Negative Recall.
- **Error Analysis**
    - **Variation** – investigate if particular models encountered errors with particular topics or tasks. This might be visible between different datasets (for example, perhaps a model doesn't generalize well to long texts).
    - **Confusion Matrix** – This would make sense to use if can accurately tag topics and zero in on whether particular topics are misinterpreted consistently/identify where a model can be improved/ is inherently weak (maybe several attempts do not improve the model).
- **Qualitative Investigation of Misinterpreted Classes**
    - user purpose – questions/requests mistaken as compliment/criticism?
    - Semantics/sentence structure/subtle language cues – jokes, sarcasm, rhetoric and literary devices
    - Atypical word usage like swearing used colloquially as a compliment
    - Errors due to mixed/conflicted sentiment
    - Discussing multiple topics (potentially with mixed sentiment)

7.1 Programming Language and Libraries for Solution Development

- **Python 3**
- **scikit-learn** – Python Machine Learning, preprocessing, pipeline and modelling tools.
- **NLTK** – Natural Language Toolkit used for NLP tasks and resources.
- **contractions** – Python NLP module for expanding contractions
- **inflect** – for generating plurals, singular nouns, ordinals, indefinite articles; convert numbers to words
- **BeautifulSoup** – for scraping, dissecting documents, dealing with data encodings, and parsers for XML/HTML.
- **TensorFlow** – Google's open source library for deep learning
- **Gensim** – fast, scalable language processing in Python

7.2 Machine Learning Design - proposed workflow for designing a solution

1. Data Collection & Assembly
2. Data Preprocessing/ text vectorization (clean and extract numerical features from text)
    a. Noise removal and writing helper functions - remove file headers & footers, strip HTML, XML, markup and metadata (I have a concern that models will learn by metadata rather than the text), and extract text from JSON-formatted data
    b. Tokenization (chunks to sentences, sentences to words, etc.)

   c. Normalization & reducing features from tokenized words
      i. Remove non-ASCII chars
      ii. Convert all characters to lower case
      iii. Remove punctuation
      iv. Replace numbers with words
      v. Remove stopwords (language plumbing)
      vi. Weighting the importance of tokens that occur in majority of samples/documents
   d. Stemming (and Lemmatization (keeps more context but expecting to be slower)
   e. Build dictionaries - Words and frequency counts, N-grams (varying lengths)
3. Data Exploration & Visualization
4. Model Building
   a. Baseline Models - Naïve Model - guessing majority class, baseline of Default Settings Logistic Regression on Bag-of-Words (only uni-grams)
   b. Default Settings Logistic Regression Using TFIDF (weighting) scale down impact of very frequent tokens versus those occurring in a smaller fraction of documents Ensemble models
   c. Feature engineering
      i. N-grams
         1. Logistic Regression with Bag of Words adding n-grams to capture some context
         2. Logistic Regression with TF-IDF adding n-grams
      ii. Grid search combination of models - attempt to recapture lost information from text pre-processing using presence and type of Characters, Punctuation, Digits, Unusual Words/Slang, Diversity of Text, Fraction of Stopwords, Average Word Length and Sentence Length
   d. word2vec – out of box (would train on corpus if had time)
5. Model Evaluation
   a. Examine model performance regarding data and tasks
   b. Comparison between models, contrasting trade offs (strengths/weaknesses) for consideration when contemplating applications (constraints like computational resources and time to compute/size of data vs good enough)
   c. Gathering self-labeled data from UrbanDictionary.com to anecdotally test models

References

Sentiment:

- [pdf] Mantyla, M., Graziotin, D., & Kuutila, M. (2016). The Evolution of Sentiment Analysis – A Review of Research Topics, Venues, and Top Cited Papers.

Interpretable Machine Learning

- [pdf] Finale, Doshi-Velez, & Been Kim (2017). Towards A Rigorous Science of Interpretable Machine Learning.

Introduction of the LSTM model:

- [pdf] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Addition of the forget gate to the LSTM model:

- [pdf] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

More recent LSTM paper:

- [pdf] Graves, Alex. Supervised sequence labelling with recurrent neural networks. Vol. 385. Springer, 2012.

Benchmarking Sentiment Analysis

- [pdf] Abbasi, Ahmed & Hassan, Ammar & Dhar, Milan. (2014). Benchmarking Twitter Sentiment Analysis Tools. Language Resources and Evaluation Conference.
- Kurtulmus, Besir. "Benchmarking Sentiment Analysis Algorithms." Algorithmia, January 18, 2016, https://blog.algorithmia.com/benchmarking-sentiment-analysis-algorithms/.

- Mannes, John. “OpenAI sets benchmark for sentiment analysis using an efficient mLSTM.” TechCrunch, April 7, 2017, https://techcrunch.com/2017/04/07/openai-sets-benchmark-for-sentiment-analysis-using-an-efficient-mlstm/.