Datametica - Text classification

Saturday, September 9, 2023 3:11 PM

Step	Description
Problem Definition	Topic modeling on customer feedback can be immensely valuable, allowing you to discover underlying patterns and themes in your customer complaints, which can help drive process improvements. I'll guide you through the steps on how to approach this:
1. Data Collection	Gathering data from various sources - Customer support channels • (e.g., ticketing system) • Web scraping tools for social media • Beautiful Soup • Scrapy • Tweepy for Twitter • Speech-to-text tools • Google's Speech-to-Text API • IBM's Watson

2. Text Preprocessing	Cleaning and preparing the text data for analysis -
	• Text normalization
	  • Stop words removal
	  • Tokenization
	  • Stemming
	  • Lemmatization
	  • Case - Folding
	• Text standardization
	  • Jargon removal - defaulter, bail-out
	  • Removing grammatical mistakes
	• Part-of-Speech tagging
	• Named Entity Recognition
	• Dependency parsing

3. Text Analysis similar to EDA	Analyzing Word Distributions and Relationships:
	• Word Distributions:
	  • Refers to the frequency or probability distribution of words appearing in a text corpus.
	  • Often visualized using histograms or bar plots to identify the most common words or phrases in a given dataset.
	• N-gram Analysis:
	  • N-grams are continuous sequences of n items (words, characters, etc.) from a text.
	  • Helps in understanding context and capturing phrases. E.g., "New York" would be a 2-gram (bigram).
	• Sparsity Visualization:
	  • Often related to the term-document matrix, where many entries are zero because a large vocabulary results in many words not appearing in individual documents.

	• Visualization can be done using a matrix plot, highlighting which terms appear in which documents.

Count Vectorization	• Count Vectorizer:
	  • Converts a collection of text documents to a matrix of word/token counts.
	  • The result is a sparse matrix where rows are documents and columns represent word counts.
	• TF-IDF (Term Frequency-Inverse Document Frequency):
	  • Similar to the Count Vectorizer, but instead of raw counts, the matrix represents the weighted count where weights are determined by the importance of a word to a document relative to its frequency across all documents.

Word Embeddings	• A form of representing words as dense vectors such that words with similar meanings are close to each other in the vector space.
	• Result in dense vectors, typically with hundreds of dimensions.
	• Word2Vec
	  ○ Developed by Google, it captures semantic relationships between words. Has two architectures: Skip-gram and Continuous Bag of Words (CBOW).
	• GloVe (Global Vectors for Word Representation)
	  ○ Developed by Stanford, it focuses on word co-occurrence statistics.
	• FastText
	  ○ Developed by Facebook, it represents words as bags of character n-grams, allowing it to handle out-of-vocabulary words.

Contextualized Word Embeddings	More advanced than traditional word embeddings.
	They consider the context around a word, allowing the same word to have different vectors based on its surrounding words.
	• a. ELMo (Embeddings from Language Models):
	  ○ Developed by AllenNLP, it uses character-based word representations and bidirectional LSTMs.
	• b. BERT (Bidirectional Encoder Representations from Transformers)
	  ○ Developed by Google, it's a transformer-based model pre-trained on a large corpus, known for its bidirectionality.
	• c. GPT (Generative Pre-trained Transformer)
	  ○ Developed by OpenAI, it's a transformer model trained to predict the next word in a sequence.

ML Models	• Naïve Bayes • Logistic Regression • Decision Trees • Random Forest • XGBoost

DL Models	• RNN
	• LSTM
	  • Regularization
	  • Early Stoppings
	  • Skip Connections

Fine Tuning	• Cross Validation

	• Hyper-parameter Tuning
Metrics	F1 score, Accuracy, Precision, Recall.
Exception Handling	• Human in the Loop • Feedback Loop

After Developing the model	Step	Description
1	Model Interpretation	Understand the patterns the model has learned. Extract insights such as feature importance or coefficients, or use techniques like LIME for deep learning models.
2	Model Optimization	Based on performance, consider hyperparameter tuning (grid search, random search, Bayesian optimization). Explore feature engineering for better accuracy.
3	Deployment	Make the model available for real-world, production use. This can involve integrating it into an existing system, wrapping it in an API, or deploying it on cloud platforms.
4	Monitoring and Maintenance	Continuously monitor the model post-deployment. Set up logging and monitoring for performance. Consider periodic re-training with new data.
5	Feedback Loop	Establish a method to gather feedback on predictions. Use feedback and new labeled data for model refinement and retraining.
6	Documentation	Document every step, from data preprocessing to deployment. Ensure clarity for any person or team member reviewing or taking over the project.
7	Communication /Presentation	Prepare a presentation for stakeholders or clients. Showcase findings, performance, business implications, and future steps. Use visual aids and simple language.

Text EDA for Supervised Classification

- Text Exploratory Data Analysis (EDA) is crucial for understanding the characteristics and patterns within the data, regardless of whether the modeling task is supervised or unsupervised.
- However, with supervised classification, the emphasis shifts towards understanding relationships between the text data and the target labels, as well as understanding the distribution of the labels themselves.

EDA steps for Supervised TC	EDA Technique	Description
1	Label Distribution	Understand if the dataset is imbalanced. Visualize with bar/pie charts.
2	Text Length Distribution	Examine the distribution of text lengths. Inform preprocessing decisions and potential neural network architectures.
3	Most/Least Frequent Words	Check the most and least frequent words for each label to see which words are indicative of a certain class.

4	Word Clouds for Each Label	Visual representation of the most frequent words for each label.
5	N-gram Analysis	Investigate which combinations of words (bigrams, trigrams, etc.) frequently appear.
6	Term Frequency-Inverse Document Frequency (TF-IDF) Values	Identify high TF-IDF value words for each label to discern important but less common words for a class.
7	Correlation Between Features	Analyze how features (like TF-IDF or Count Vectorized features) correlate with each other.
8	Named Entity Recognition (NER)	Recognize named entities in the text that might be crucial depending on the domain.
9	POS (Part-of-Speech) Tagging Analysis	Understand the distribution of different parts of speech across classes.
10	Outliers Detection	Detect anomalies or outliers in text length, word usage, etc., and decide how they should be handled.
11	Sentiment Analysis	Perform a preliminary sentiment analysis for datasets where context can be seen as positive, negative, or neutral.
12	Sample Text Analysis	Randomly sample and manually inspect text for each label for a qualitative understanding of the dataset.

Word Vectorizations -- A

- *Discrete*
- Bag of Words - Count Vectorizer
- TF-IDF

Word Vectorizations -- B

- *Continuous*
- Word2Vec
- Glove
- FastText

Word Vectorizations -- C

- *Contextual*
- ELMO
- BERT
- GPT

PROs and CONs of Word Embeddings--A

Feature/Model	Word2Vec	GloVe	FastText
Methodology	Predictive model (uses context to predict word)	Count-based model (uses word co-occurrence stats)	Predictive model (similar to Word2Vec but uses subword information)
Pros			
Contextual Info	Captures semantic meaning well based on context	Captures semantic meaning from co-occurrence	Captures semantic meaning and morphological structure
Handling OOV Words	Does not handle out-of-vocabulary words	Does not handle out-of-vocabulary words	Can generate vectors for OOV words due to subword information
Training Time	Faster with negative sampling	Directly addresses the issue of word analogy	Generally slower due to subword information
Memory Usage	Requires less memory during training	Requires storing word co-occurrence matrix, can be memory-intensive	Requires more memory to store n-gram vectors
Subword Information	Does not consider subword information	Does not consider subword information	Considers subword information (n-grams)
Word Analogies	Performs well	Performs well	Performs well, especially with morphologically rich languages
Cons			
Interpretability	Internal workings can be	Large co-occurrence matrix can be hard to	Subword approach might be hard to

	hard to interpret	interpret	interpret for some applications
Training Complexity	Requires careful parameter tuning	Requires matrix factorization which can be computationally intensive	Requires more training data to cover various word forms
Size of Embeddings	Fixed for a given model	Fixed for a given model	Larger due to additional subword embeddings

PROs and CONs of Word Embeddings--B

Feature/Model	ELMo	BERT	GPT
Architecture	BiLSTM based on characters	Transformer based on words/tokens	Transformer with a generative objective
Pros			
Contextual Embeddings	First to produce embeddings with context	Deep bidirectional context	Unidirectional context (GPT-2 introduced bidirectional context)
Pre-training Data	Unlabeled text	Large amount of unlabeled text	Books, websites, and other diverse web text
Fine-tuning	Designed to be added to existing models	End-to-end training for downstream tasks	Generative tasks or fine-tuning for classification
Model Variants	Single architecture	Multiple sizes (Base, Large, etc.)	Multiple versions (GPT, GPT-2, GPT-3)
Versatility	Needs task-specific architectures during fine-tuning	One model for various tasks	Can be used for a variety of tasks without task-specific heads

Cons			
Training Complexity	Moderate (biLSTM is less complex than Transformer)	Requires significant computational resources due to the Transformer's architecture	Requires significant computational resources and data
Interpretability	Deep models can be hard to interpret	Deep models can be hard to interpret	Deep models can be hard to interpret
Size	Smaller compared to BERT & GPT	Large models can have billions of parameters	GPT-3, for instance, has 175 billion parameters
Tokenization	Character-based	WordPiece	Byte Pair Encoding (BPE)

==========

END

==========