**Contents**

[Abstract 2](#_Toc57119972)

[Introduction 2](#_Toc57119973)

[Commercial User of NLP 2](#_Toc57119974)

[Stance Classification 3](#_Toc57119975)

[SMOTE for Data Bias Elimination 4](#_Toc57119976)

[RNN with LSTM Finding Contextual Reference 5](#_Toc57119977)

[Finding Sentiment with LDA 6](#_Toc57119978)

[Analyzing Twitter Posts 7](#_Toc57119979)

[Fake News Detection LIAR 8](#_Toc57119980)

[Fake News Detection Flaws 9](#_Toc57119981)

[NLP Fake New Detection 10](#_Toc57119982)

[Fake News Labeling 11](#_Toc57119983)

[NLP Deep Learning 11](#_Toc57119984)

[The SuperGLUE Benchmark Suite 12](#_Toc57119985)

[RoBERTa 14](#_Toc57119986)

[PET and iPET 14](#_Toc57119987)

[How PET and iPET can Outperform GPT-3 15](#_Toc57119988)

[Machine Learning Cloud Platforms 16](#_Toc57119989)

[Conclusion 17](#_Toc57119990)

[References 19](#_Toc57119991)

# Abstract

Consumer sentiment can help companies understand people’s opinions about commercial products and better their organizations. Recurrent Neural Networks (RNNs), a type of DL model, are good at retaining results from past calculations, making them useful for processing sequential data like sentences and paragraphs and capturing an article’s overall meaning and context from a stream of words (Deepak and Chitturi). RNNs have been used to automatically determine sentiment from written text in venues like Google reviews and Twitter posts, as well as identify misleading sources of information, including so-called “fake news”. The best type of data for RNNs to use is sequential data. Language is a specific example of sequential data. NLP improves prediction accuracy when implemented with deep learning techniques.

LSTM (Long-Short Term Memory) models, a type of RNN, will be used to analyze reviews from three healthcare systems to determine if there is a correlation in sentiment between the healthcare systems.

# Introduction

Natural Language Processing (NLP) is a genre of Machine Learning (ML) that allows humans to teach computers how to read and interpret natural language. Machines must be taught how to understand the complex structure of natural language, including emotional connotations of text like sarcasm, lies, humor, optimism, or malcontent. Machines learn these nuances by humans training them with datasets that are labeled as such.

NLP can be used to assess consumer response to products in online forums, like Twitter, Google, and Amazon reviews. Sentiment analysis, also known as opinion mining, analyzes the large amounts of data produced by these online forums to better understand people’s opinions. Companies can then use this information to adjust their products to provide a better experience for the customers.

# Commercial User of NLP

Natural language processing (NLP) can be divided into two subfields (Dale). One, natural language understanding (NLU) attempts to interpret contextual meaning. According to Dale, for NLU technology to be practical for business applications, NLU must identify a document’s context and incorporate that into its interpretation. The other, natural language generation (NLG), attempts to predict words or phrases that would follow from an initial phrase in a given context. NLG can make useful predictions of speech whenever it can identify the context of an individual textual reference.

Currently, the most notable business applications of NLG include translating text between languages, fully autonomous article generation, and augmented text generation – i.e., predicting what words a user will type next. These applications have succeeded to varying degrees, depending in part upon their accuracy, which correlates strongly with broader market viability.

To date, the least successful business model is fully autonomous article generation. Generating a fully autonomous article in a GPT-2 model requires a preliminary training process. Training a model requires a large dataset on a specific topic. Generating large datasets can be expensive and the GPT-2 model is not advanced enough to grasp certain bias information about a topic. The model is better at generating general content about non-technical topics. Additionally, gaps in a typical GPT-2 model’s knowledge will hinder it from generating a fully believable article.

Two more effective business models for NLG are language translation and writer-assisted augmentation. NLG can successfully translate meaning between different languages. NLG is also competent at word suggestion, as evidenced by Google’s SmartCompose, which is integrated into Google’s Gmail application. SmartCompose predicts how users might respond to email messages and offers suggestions as users enter text. Email text prediction is an appropriate scenario for NLG because it allows the GPT-2 model to be trained for specific, granular use-cases.

# Stance Classification

Stance classification attempts to classify a writer’s position about a topic. Instance classification, probabilistic, hidden Markov, and DL models are used to determine if an article's body correlates with its title; if not, a flag is raised for fake news. Other means of classifying stance include long short-term memory (LSTM) and neural networks.

LSTM (long short-term memory) and GRU (gated recurrent unit) models are a good fit for interpreting text and voice. Both are suitable for processing long inputs such as news articles. GRUs have fewer gates to manage and produce approximately the same results as LSTM, eliminating complexity and additional work. The flow through each of a GRU’s gates is determined by a number between 0 (no flow) and 1 (wide open) (Olah). Gates allow cell states to change based upon data being passed through the gate. The model then evaluates each cell state to determine which cell states to throw away, also known as training the model.

# SMOTE for Data Bias Elimination

Data sets can inherently have unbalanced, biased data. To correct imbalances in data sets, data scientists use oversampling and undersampling to create an even distribution of data.

Oversampling detects minority datapoints and duplicates them to balance a data set. SMOTE (Synthetic Minority Oversampling Technique), a preferred method for oversampling, provides extra datapoints without adding new data points to a data set. Adding new data points can change how a model will be trained and cause inconsistent results (Brown). SMOTE, however, fails to account for majority classification datapoints when augmenting minority datapoints. Failures to observe a strong overlap between the two can produce ambiguous data points.

Edited Nearest Neighbor (ENN) undersampling finds datapoints classified in minorities that should be classified in the majority (Brownlee). Misclassified points are then removed, preserving correct classifications. A popular approach for extending random-sampling ENN is Tomek Links. Tomek Links fixes the irregularities associated with random sampling by finding the cross-class pairs that define the classification boundaries (see Figure 1). For best results, Kudugunta et al. recommend using ENN and then the Tomek Links method. ENN finds misclassifications that are deep in the wrong set and Tomek Links cleans up by attending to the granular boundary details.

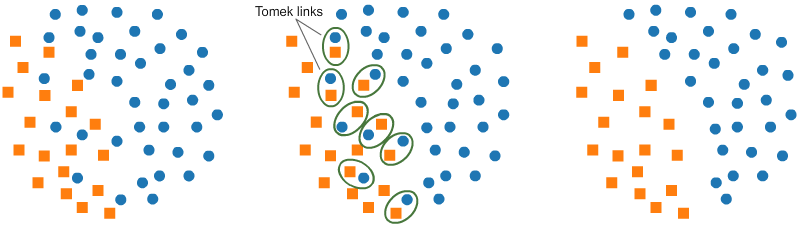


Figure 1 Tomek Links Correctly Classifying Datapoints (Alencar)

SMOTE combined with Edited Nearest Neighbors (ENN) or “Tomek Links” can be used to classify data sets with granularity: SMOTE balances the data and ENN plus Tomek Links cleans up misclassifications. Each combination improves classifications in different ways. For best results, Kudugunta et al. recommend using both combinations to classify data sets – SMOTE plus ENN (SMOTENN) and SMOTE plus Tomek Links (ibid.).

# RNN with LSTM Finding Contextual Reference

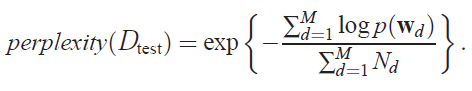
Recurrent Neural Networks (RNN) are good at remembering short-term information (Olah). Unfortunately, when RNNs are used to analyze a large text corpus, they have trouble preserving contextual reference by the end of the text. Preserving contextual reference requires an advanced neural network like the Long Short-term Memory (LSTM) network. Hochreiter and Schmidhuber introduced LSTM as a “novel recurrent network architecture in conjunction with an appropriate gradient-based learning algorithm.” (Hochreiter and Schmidhuber) LSTM corrects error back-flow problems, learning to bridge time intervals with no loss of time lagging capabilities. It combines short and long-term memory architectures, making LSTM a good tool for finding relationships in sequential data, e.g. text or Tweets.

If word embeddings are combined with LSTM networks, vectors can assimilate additional word representations. Pre-training a set of Global Vectors for Word Representation (GLoVE) is an appropriate strategy for finding contextual Twitter data. Data must be preprocessed before feeding it to an LSTM. Before GLoVEs can be trained, Twitter data needs to be preprocessed by replacing hashtags, URLs, numbers and user mentions with tags: “<hashtag>,” “<url>,” “<number>,” or “<user>.” The same approach is used for emojis, i.e. “<smile>,” “<laugh>,” or “<cry>”. Words in all uppercase,are converted to lowercase, followed by the tag, “<allcaps>”.

Random Forest and AdaBoost Classifiers are the best options for classifying Twitter bots that are detected with account-level data. AdaBoost paired with SMOTENN (SMOTE plus ENN) yielded an accuracy rate of 0.9931. Detecting individual Tweets from Twitter bots needs more data to combine with metadata for improved results unless using oversampling with ENN. LSTM models are the best for detecting Tweets written by bots because of their ability to extend vectorization with word embeddings.

# Finding Sentiment with LDA

Latent Dirichlet allocation (LDA) allows for advanced data visualization (Liu). LDA analyzes text in documents, derives contextual meaning from a corpus, and can assimilate by using a multinomial probability conditioned on a topic (Blei, Ng and Jordan). In short, presenting an LDA with more topics reduces its perplexity (Figures 1 and 2). According to Blei et al., “topic distributions can be conditioned on features like *paragraphs* or *sentences.”,* providing a precise means of contextual analysis.



Equation 1 Perplexity for M documents

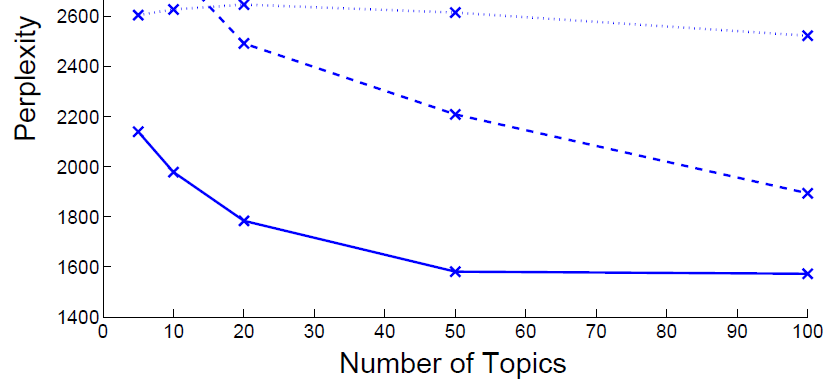


Figure 2 Increasing topics decreases perplexity

When combined with LDAT, sentiment analysis can be used to scrutinize unstructured textual data. Sentiment analysis can extract opinions from the noisy text, which allows explorations of how consumers regard brands and how sentiments vary among brands within and across industries (Liu). Twitter is inherently noisy, due to its structure, and produces massive amounts of unstructured data.

Sentiments expressed in Tweets can be analyzed with the ML maximum entropy classifier. Ratnaparkhi states, “Maximum entropy models offer a clean way to combine diverse pieces of contextual evidence to estimate the probability of a certain linguistic class occurring with a certain linguistic context.” (Ratnaparkhi) Maximum entropy can be used to quantify unstructured textual big data. Before feeding data to a maximum entropy model Tweets need to be labeled as positive, neutral, or negative. LDA can also reveal what combinations of products consumers like to purchase.

# Analyzing Twitter Posts

Twitter posts are limited in character size and thus are often brief and informal. They regularly contain out-of-vocabulary terms (OOVs) such as slang, typographical errors, abbreviations, and incorrect grammar (Imran, Mitra and Castillo). NLP models use classifiers to identify OOV terms, characterizing them as typos/misspellings, single-word abbreviations/slangs, multi-word abbreviations/slangs, phonetics substitutions, or words without spaces. These lexical variations are used to create an auxiliary vocabulary, which is tested against large dictionaries to validate their entries’ statuses as OOV words. For example, cities, particularly international cities, are often incorrectly labeled as OOVs. This is corrected by comparing OOVs with a city dictionary (MaxMind).

OOVs, once categorized, are annotated with the correct word and meaning. Language models can be trained to associate misspelled words with their proper spellings using a dictionary of popular words like Wiktionary or SCOWL. Conditional probability is used to find misspellings. Imran et al. explain the misspelling equation as, “for each misspelled word *w,* find a correction *c* out of all possible corrections where the probability [*P*]of *c* given *w* is maximum.” (ibid.) These parameters define what is needed to find the probability of an event using Bayes Theorem: *argmaxcP(c|w) = argmaxcP(w|c)P(c)*.

Determining an OOVs message type requires classification algorithms. These algorithms include Naïve Bayes (NB), Support Vector Machines (SVM), and Random Forest (RF). These classifications were used to classify messages from over fifty-two million crisis labeled Tweets, tweeted during nineteen crisis events.

Bigram models are used to predict word choice based on distribution representation. Word2vec, a common bigram, is used to train word embeddings; these map words to vectors instead of numeric values. The training process replaces stop-words, URLs, digits, and usernames with a fixed constant and removes special characters. Generating word embeddings can use Continuous Bag of Words (CBOW) architecture with a negative sampling coupled with word representation dimensionality.

# Fake News Detection LIAR

Agarwal et al. compared the effectiveness of Naïve Bayes, logistic regression, linear SVM, stochastic gradient, and RF classifiers for fake news detection. The authors’ datasets, which were sourced from LIAR[[1]](#footnote-1), contain 13 variables/columns for training, testing, and set validation.

Agarwal et al. limited their study to two variables: statement, a news article’s body of text, and label, the classification of the article’s truthfulness. In the original LIAR dataset, labels were split into true and false columns with three distinctions per column:true, mostly-true, and half-true for the one and barely-true, false, and pants-fire for the other. For simplicity, the authors treated all *true* options as true and all *false* options as false.

Agarwal et al’s procedure for feature extraction began by removing stopwords, whitespace, and punctuation; according to the authors, these features provide no contextual meaning and hinder performance. The procedure then identified the core meanings of words, using a process called lemmatization. Finally, N-grams were applied to a text to find contextual references. This was done to help predict what words should appear in a corpus. These preprocessing steps produced a set of features that can be fed into multiple classifiers for fake news detection.

The authors’ classifiers were obtained from a Python library, Sci-kit Learn. The library supports the preprocessing of content in a form that these classifiers accept. Each classifier fits the data to its model. After fitting the classifiers, the authors used a method (classification parameter) in the library’s GridSearchCV class to determine the classifier that produces the best result. For the authors’ dataset, the best performing model was the SVM classifier.

# Fake News Detection Flaws

NLP algorithms for detecting fake news have flaws that attacks can exploit. Current NLP algorithms check a variety of data points from news article to determine if it is fake. One method of fake news detection compares an article’s headline text with its body text, checking for correlation: i.e. “is the article *clickbait?*” (Zhou, Guan and Bhat) Articles are checked for biased or inflammatory wording. Unfortunately, authors of fake news articles can avoid these detection methods, e.g., by making slight changes to an authentic news article that slants the article in a specific way.

Adversarial machine learning attempts to understand how malicious users can defeat machine learning classifiers. Types of attacks include fact distortion, subject-object exchange, and cause confounding. Fact distortion attacks are exaggerations that modify or distort the original article. Subject-object exchange is an attack that attempts to confuse readers so they do not understand who acts or receives an action. A cause-confounding attack either creates a non-existent connection between independent events or removes part of the story. These three types of attacks are not detected by normal fake news classifiers.

Zhou et al. created a model, Fakebox, that detects advanced forms of fake news articles. Fakebox, like other models, needed a proper dataset to classify fake news – i.e., to learn what fake news looks like. Zhou et al. trained Fakebox using McIntire’s *fake-real-news-dataset*. This is an open-source dataset, used predominately for misinformation research, which contains passages from and titles of articles.[[2]](#footnote-2)[[3]](#footnote-3)

Fakebox determines an article’s authenticity from its linguistic characteristics, including the article’s title, content, and URL—the latter of which, in the case of McIntire’s dataset, was missing. Fakebox analyzes titles and headlines checking content for coherence and style; determining if the title and body text correlate; and checking article domains (clickable links) for authenticity and reputation.

Fakebox labels dataset articles as either written in a real-news form, unlike real news form, or unsure. It assigns scores of 60 to 100 to articles “written in a real-news form”, 0-40 to articles “unlike real-news form”, and 40- 60 to others.

# NLP Fake New Detection

Two common NLP-based mechanisms for detecting fake news are classification and regression testing ( Oshikawa, Qian and Wang). Classification, the more popular mechanism, has a high percentage of success when limited to a binary set – i.e. real or fake news. Problems arise with binary classification because fake news articles tend to combine truthful statements and misleading or false statements.

One alternative to binary classification uses additional categories to classify articles. This approach, however, can yield less accurate classifications. Alternatively, regression testing can be used to rate truthfulness on a numeric scale. Regression testing requires that datasets be assigned numeric scores. A numeric scale can be calculated from discrete labels of truthfulness using the difference between predicted scores and ground truth scores based on Pearson/Spearman correlations. Ground truth scores are predefined because the outputs are known.

Mechanisms for classifying fake news include non-neural network models, neural network models, rhetorical approach models, and RTE-based textual entailment recognition. Before generating a model from a dataset, the dataset is usually preprocessed using tokenization, stemming, and generalization for weighting words. Algorithms like Term Frequency-Inverse Document Frequency (TF-IDF) and Linguistic Inquiry and Word Count (LIWC) are used to analyze a text’s content; TF-IDF rates the importance of a text’s words and LIWC characterizes words’ psychological meanings. Word sequences can also be tokenized. Oshikawa et al. cite Word2vec as a useful tool for transforming word sequences into features in a dataset, useful for training a model.

Typically, the most challenging part of automated fake news detection is finding quality datasets. Datasets typically must be annotated by humans before being preprocessed. This is a challenge because large datasets are typically used to train models. One alternative to the use of large datasets is to judge articles based on news source metadata, such as author, news outlet, date, and URL. While metadata can be authoritative, biased classification processes could label stories from small news outlets as fake based upon their lack of popularity instead of their truthfulness.

# Fake News Labeling

Supervised and semi-supervised models are trained using labeled data sets. Labels enable computers to understand what fake news looks like. One way to label data is to have a human randomly sample data from a data set and classify sampled data appropriately. This form of labeling is time-consuming but produces good data to train models.

A contextual LSTM architecture can use a Tweet’s content and metadata for bot detection (Kudugunta and Ferrara). Examining additional features such as account metadata, network structure information, and temporal activity increases detection accuracy. However, fewer features make ML models more efficient, allowing them to be trained faster and easier while rendering them less prone to overfitting.

Methods for detecting Twitter bots include account level and Tweet level detection. Account-level detection uses metadata associated with a Tweeter’s account; it requires a minimal set of features with little to no preprocessing; RF classifier is the best out-of-the-box account-level approach. Tweet level detection uses a Tweet’s text, along with additional metadata; the latteris required for correct classification.

# NLP Deep Learning

Deep learning (DL), a subfield of machine learning (ML), focuses on learning data representation instead of task-specific algorithms. DL learning types include supervised, semi-supervised, and unsupervised learning. These differ based on the availability of *ground truth*: a preexisting definition of the model’s expected output (Soni). Supervised learning generates models from a sample of data, based on users’ definitions of a model’s expected outputs. Semi-supervised learning generates models from a mix of labeled and unlabeled data; it can automatically label the unlabeled data. Unsupervised learning accepts data in its natural form; there are no output labels, just structured data. Unlike supervised and semi-supervised where most or all data must be labeled, unsupervised data does not need labels.

Natural language processing (NLP) and data mining use deep learning architectures to detect fake news (Deepak and Chitturi). NLP improves prediction accuracy when implemented with deep learning techniques. Recurring neural networks (RNN), a type of DL model, are good at retaining results from past calculations, making them useful for processing sequential data like sentences and paragraphs and capturing an article’s overall meaning and context from a stream of words (ibid.).

An article's context is insufficient for determining its truth. Determining if articles are fake news requires external data sources, gathered from the web. Scraped web data – what Deepak and Chitturi call “live data mining” – combined with existing metadata adds information for truth detection. Live data mining is the use of an article’s word tokens to query search engines for related data to scrape. Query results are then validated using cosine similarity to evaluate titles for consistency. For every word token, this process is used to find additional metadata. The additional metadata is then concatenated to the original article, preparing the data set for word vectorization, structuring the data set to be trained with DL architecture.

# The SuperGLUE Benchmark Suite

The SuperGLUE benchmark suite is designed to evaluate how an NLP model performs on a broad range of language understanding tasks (Wang, Pruksachatkun and Nangia). Multiple groups of researchers maintain top benchmarking results on SuperGLUE’s website.[[4]](#footnote-4)[[5]](#footnote-5) SuperGLUE evaluates models according to the following criteria:

* *task substance* - the ability to understand and reason texts written in English
* *task difficulty* - the extent to which a task is beyond the scope of current state-of-the-art systems, but solvable for college-educated English speakers, excluding domain-specific knowledge such as medical notes
* *evaluability* – the existence of automatic performance metrics corresponding to human judgments of output quality

SuperGLUE requires models to use *existing* public data for benchmarking so anyone can view the dataset for authenticity. Datasets need to be formatted for simple inputs and outputs to prevent the creation of complex task-specific model architectures. Data must also be publicly accessible and licensed for use by other researchers.

GLUE, SuperGLUE’s predecessor, was also designed to provide a general-purpose evaluation of language understanding. GLUE’s measurement objectives that are still relevant are training data volumes, task genres, and task formulations. However, GLUE’s high-level goals have become obsolete due to new model sizes, model structure, and advances in NLP techniques.

To compensate for GLUE’s shortcomings, SuperGLUE added more challenging and diverse task formats; comprehensive human baselines, i.e. a measurement showing where machines stand against humans; improved code support, i.e. “… a modular toolkit for work[ing] on pretraining, multi-task learning, and transfer learning in NLP, built around standard tools including PyTorch” (ibid.); and refined usage rules, e.g. public datasets and open use license for datasets. Aspects of GLUE that SuperGLUE retained included a public leaderboard; a focus on benchmarking based on existing data, accompanied by single-number performance metrics; an analysis toolkit; and benchmarking based on eight language understanding tasks. These tasks include Boolean Questions, CommitmentBank, Choice of Plausible Alternatives, Multi-Sentence Reading Comprehension, Reading Comprehension with Commonsense Reasoning Dataset, Recognizing Textual Entailment, Word-in-Context, and Winograd Schema Challenge) Each task provides a separate set of challenges for NLP methods to address.

Top-performing NLP models on SuperGLUE’s website included ELMo, OpenAI’s GPT, and BERT. These NLP methods do self-supervised learning: they learn from massive unlabeled corpora with formulas that adapt to modeling targeted tasks. They also excel at doing series of tasks: e.g., answering questions, sentiment analysis, textual entailment, and parsing.

# RoBERTa

In their 2019 paper, Liu et al describe RoBERTa, “A robustly optimized BERT pretraining approach.” RoBERTa is an improved version of BERT, a language model designed to understand natural language structure (Liu, Ott and Goyal). Liu et al. improved BERT by first training BERT with bigger batches and more data. They then removed BERT’s focus on next sentence prediction and trained it on longer sequences of data. This changes the masking pattern and applies the changes to the training data.

To train BERT, Liu et al. used two segments of inputs and sequences of tokens. Each training segment usually consisted of two or more sentences. The two segments were concatenated and presented to BERT as a single input sequence. In the first segment, BERT was pre-trained with a large text corpus that is an unlabeled dataset. Then, BERT was fine-tuned using end-task labeled data as required by its architecture. This architecture, Vaswani et al.’s Transformer, is “based solely on attention mechanisms, dispensing with recurrence and convolutions entirely” (Vaswani, Shazeer and Parmar). The Transformer architecture enables BERT to be trained in less time with better results.

Because of its structure, BERT requires an enormous amount of data to pre-train. To get adequate data Liu et al. used multiple sources for datasets: BookCorpus, plus the English Wikipedia, CC-News, OpenWebText, and Stories. Combining these datasets allowed Liu et al. to successfully train RoBERTa.

# PET and iPET

PET (Pattern-Exploiting Training) trains models using cloze questions as inputs (Schick and Schutze, Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference). A cloze question is a text that contains a question, together with various answers to that question, embedded in the text (https://wp.stolaf.edu/it/embedded-answers-cloze-questions/). Because cloze questions provide their answers, PET accepts unlabeled datasets, which allows for semi-supervised learning. PET and iPET can be trained with little to no data and perform better than the GPT-3: a high performing, pre-trained language model that can generate coherent human-like text.

PET must learn from each new example it accepts, making PET difficult to use with small datasets. Plausible justification, the ability to trust a variable’s (word token’s) context, is difficult to achieve with small datasets: small datasets cannot provide enough context to identify contextual references. PET, however, can gather enough context from small datasets to infer when a text is referencing, directly or indirectly, a specific word.

A second disadvantage of PET is that the “distilling of pattern-based models into a single classifier deprives individual models of an opportunity to learn from each other.” (ibid.) This limitation is addressed by iPET, an iterative variant of PET that trains various generations of models on datasets of increasing size. *Figure 2* illustrates that datasets are enriched with original data 𝒯 from examples 𝒟 that are labeled using previous-generation models. Iterations improve iPET’s understanding of contextual reference when predicting what is to come next.

# How PET and iPET can Outperform GPT-3

Studies by Schick and Schutze indicate that PET and iPET can outperform GPT-3 with fewer parameters (ibid.). GPT-3 relies on priming: GPT-3 must be provided with examples of inputs and corresponding outputs as the context for predictions. No gradient update is performed. GPT-3 requires a huge learning model and can only scale to a few examples. In part, GPT-3’s inability to scale is due to its dependence on large datasets. Few real-world datasets will be large enough to train GPT-3.

PET, unlike GPT-3, accepts training input as cloze questions with regular gradient-based finetuning, containing task descriptions.[[6]](#footnote-6)[[7]](#footnote-7) PET, moreover, can predict words with less unlabeled data, making it a better fit for real-world problems. However, PET can only predict words that correspond to single tokens in its vocabulary. As a result, trained models are limited to those that can be worded using single tokens in the vocabulary, i.e. the sentence.

Schick and Schultze used a combination of PET, iPET, and ALBERT-xxlarge-v2[[8]](#footnote-8) to obtain inferences with 99.9% fewer parameters than GPT-3. Their model achieved better results than GPT-3 when applied to the SuperGLUE benchmark suite. Key factors in this outcome were PET’s ability to combine multiple task (model training) formulations and its tolerance for difficult-to-understand wordings. The authors trained their model with *pattern-verbalizer pairs* (PVPs): maps from inputs to cloze questions, where each question contains a single mask with a map from each output to a single token that “represent[s] its task-specific meaning in the pattern” (ibid.).[[9]](#footnote-9) This enables PET to recognize the answer to a question within a sentence.

# Machine Learning Cloud Platforms

Machine Learning is GPU intensive due to the large size of data sets. For DL models, the Google Colab [sic] cloud platform is a good choice for accelerated GPU/TPU’s, decreasing the training process time (Deepak and Chitturi). Colab allows users to run blocks of python code independent of each other. The base Colab option is free, but Colab has faster GPU/TPU’s that are paid option. This cloud option also connects with TensorFlow, a Google Machine Learning platform that has several types of NLP models to train using Colab as a code management source.

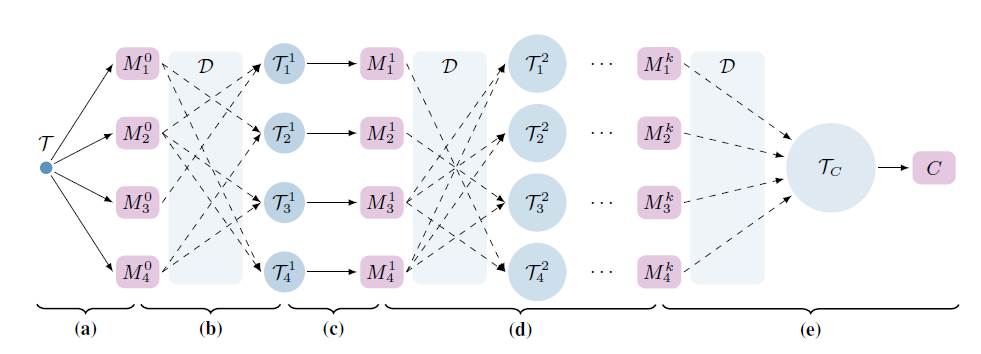


Figure 2. Illustrating iPET’s operation. **a.** initial training for PET; **b.** iPET generates a new training set from two randomly generate subsets of initial models (⋋= 2/3); **c.** a new PET model is trained using the larger model-specific datasets created in **b**; **d.** steps b and c are repeated 𝓀 times; **e.** models generated at the end of 𝓀 are used to create soft-labeled datasets as in regular PET (Schick and Schultze, op. cit..).

# Conclusion

Given a suitable dataset, NLP can determine an individual’s opinion on a subject. Opinions can be mined from Google, Bing, Yelp, or Twitter tweets. On Google, Bing, and Yelp people leave reviews about businesses and on Twitter people will often tweet about their interactions with businesses. Using these collections of data, NLP models can be trained to learn people’s opinions about companies. Researchers can then try to find correlations people’s opinions about different companies.

For my thesis, I will review data about three healthcare systems on Google, Yelp, Bing, and Twitter that were generated by customers. Gathering reviews people post about the specific healthcare systems will allow me to analyze their sentiment toward the healthcare systems. Twitter also serves as a strong platform for gathering information about how people feel toward healthcare systems. Using these four data sources, I will evaluate people’s sentiments about each healthcare system. My prediction is that each platform may produce different sentiment results. So, I will analyze each data source (Google, Yelp, Bing, and Twitter) separately and then assess deviations among the four. Using the highest correlating sentiment between the healthcare systems, I will try to determine why people may have the same sentiment toward different healthcare systems.

Each data source will provide different challenges for formatting data. Google, Yelp, Bing, and Twitter each impose a different structure on their data. To parse data from each source correctly, I will have to create an architecture that can capture the data from each platform that will be required for sentiment analysis. This new data architecture will need to support the parsing of data fro machine learning models. In addition to this new data architecture, the original structures gathered from my sources will be kept. Using the original structures I will attempt to compare sentiment between the platforms as well. However, reviews and Tweets will not be compaired. The sentiment between a review and a Tweet may have a different context and affect the outcome of comparative sentiment analysis.

The NLP model will produce what people feel a specific way toward a healthcare system. My model will classify the results into two categories: negative and positive. Within these classifications will be sub-classifications. The sub-classifications will be the subject of the review/Tweet. Each sub-classification can have multiple reviews/Tweets. I will use these classifications to analyze each of the three healthcare systems. This type of classification will produce results showing negative and positive subject matters between the systems and can be formatted into a list. The sub-categories should show if there are correlations between the systems.

# References

Oshikawa, Ray , Jing Qian and William Yang Wang. "A Survey on Natural Language Processing for Fake News Detection." *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, 2020. 6086-6093. Print.

Alencar, Rafael. *Resampling strategies for imbalanced datasets*. 2017. Web. 2 October 2020.

Blei, David M, Andrew Y Ng and Michael I Jordan. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* (2003): 993-1022. Print.

Brown, Jason. *SMOTE for Imbalanced Classification with Python*. 3 January 2020. Web. 3 October 2020.

Brownlee, Jason. *Undersampling Algorithms for Imbalanced Classification*. 20 January 2020. Web. 25 September 2020. <https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/>.

Dale, Robert. "Natural language generation: The commercial state of the art in 2020." *Natural language engineering 26.4* (2020): 481-487. Web.

Deepak and Bhadrachalam Chitturi. "Deep neural approach to Fake-News identification." *Procedia computer science 167* (2020): 2236–2243. Print.

Hochreiter, Sepp and Jürgen Schmidhuber. "Long Short-Term Memory." *Neural Computation* 9 (1997): 1735 - 1780. Print.

Imran, Muhammad, Prasenjit Mitra and Carlos Castillo. "Twitter as a Lifeline: Humban-annotated Twitter Corpora for NLP of Crisis-related Messages." *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Paris: European Language Resources Association (ELRA), 2016. 23-28.

Kudugunta, Sneha and Emilio Ferrara. "Deep neural networks for bot detection." *Information sciences 467* (2018): 312–322. Print.

Liu, Xia. "A big data approach to examining social bots on Twitter." *Journal of Services Marketing 33.4* (2019): 369–379. Print.

Liu, Yinhan, et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv* (2019): arXiv:1907.11692. Web.

Olah, Christopher. *Understanding LSTM Networks*. 27 August 2015. Web. 2 October 2020.

Ratnaparkhi, Adwait. *A Simple Introduction to Maximum Entropy*. Technical Report. Pennsylvania: University of Pennsylvania Institute for Research in Cognitive Science Technical Report, 1997. Print.

Schick, Timo and Hinrich Schutze. "Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference." *ArXiv* (2020): n. web.

Soni, Devin. *Understanding the Different Types of Machine Learning Models*. 25 August 2019. Web. 25 September 2020.

Vaswani, Ashish, et al. "Attention Is All You Need." *arXiv* (2017): arXiv:1706.03762. Web.

Wang, Alex, et al. "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems." *NeurIPS* (2019): n. Web.

Zhou, Zhixuan, et al. "Fake News Detection via NLP is Vulnerable to Adversarial Attacks." (2019).

1. https://www.cs.ucsb.edu/˜william/data/liar\_dataset.zip [↑](#footnote-ref-1)
2. <https://raw.githubusercontent.com/lutzhamel/fake-news/master/data/fake_or_real_news.csv> [↑](#footnote-ref-2)
3. Please note: Footnote two is a 28 MB file, so it typically takes about thirty seconds to download. [↑](#footnote-ref-3)
4. <https://super.gluebenchmark.com/tasks> [↑](#footnote-ref-4)
5. Machine Learning for Language at NYU, Paul G. Allen School of Computer Science & Engineering, DeepMind, Facebook AI, and Samsung Research [↑](#footnote-ref-5)
6. NLP Tasks: “A model is trained on a large number of labeled examples from which it then generalizes to unseen data” (Schick and Schultze, op. cit..). . [↑](#footnote-ref-6)
7. Task Descriptions for Neural Architectures: “…append description in natural language to an input and let the language model predict continuation that solve the task” (Schick and Schultze, op. cit..).. [↑](#footnote-ref-7)
8. ALBERT is a self-supervised learning model of language representations. Acting like a smaller version of BERT, ALBERT can be trained to understand natural language in less time with fewer parameters and better results. [↑](#footnote-ref-8)
9. “Masked language modeling is a fill-in-the-blank task, where a model uses the context words surrounding a [MASK] token to try to predict what the [MASK] word should be” (Masked Language Modeling). [↑](#footnote-ref-9)