Deep or Simple Models for Semantic Tagging? It Depends on your Data

Jinfeng Li, Yuliang Li, Xiaolan Wang, Wang-Chiew Tan
Megagon Labs
{jinfeng,yuliang,xiaolan,wangchiew}@megagon.ai

ABSTRACT

Semantic tagging, which has extensive applications in text mining, predicts whether a given piece of text conveys the meaning of a given semantic tag. The problem of semantic tagging is largely solved with supervised learning and today, deep learning models are widely perceived to be better for semantic tagging. However, there is no comprehensive study supporting the popular belief. Practitioners often have to train different types of models for each semantic tagging task to identify the best model. This process is both expensive and inefficient.

We embark on a systematic study to investigate the following question: Are deep models the best performing model for all semantic tagging tasks? To answer this question, we compare deep models against "simple models" over datasets with varying characteristics. Specifically, we select three prevalent deep models (i.e. CNN, LSTM, and BERT) and two simple models (i.e. LR and SVM), and compare their performance on the semantic tagging task over 21 datasets. Results show that the size, the label ratio, and the label cleanliness of a dataset significantly impact the quality of semantic tagging. Simple models achieve similar tagging quality to deep models on large datasets, but the runtime of simple models is much shorter. Moreover, simple models can achieve better tagging quality than deep models when targeting datasets show worse label cleanliness and/or more severe imbalance. Based on these findings, our study can systematically guide practitioners in selecting the right learning model for their semantic tagging task.

1. INTRODUCTION

A lot of applications for processing text rely on tagging words, phrases or sentences with semantically informative tags. Sentiment analysis [27, 56, 39], for example, annotates sentences or phrases with a sentiment tag that indicates whether the sentence has a positive or negative sentiment. These sentiment tags are exploited by downstream applications to determine appropriate actions. Another example is entity tagging, which determines if a span in the text refers to a real-world object. Generally speaking, the task of annotating text with semantic tags can be referred to as the semantic tagging problem. More precisely, semantic tagger takes a piece of text and a predefined tag as inputs, and outputs whether this text conveys the semantics of the tag. In this paper, we focus on short text, which can be a sentence, a paragraph, or a passage. We also refer short text loosely as sentence.

There are two types of methods for semantic tagging: rule programming and supervised learning. Rule programming-based methods require an expert to specify rules for semantic tagging. This is often error-prone and requires significant programming effort. In contrast, supervised learning models do not require much programming effort. However, training these models requires labeled data but can typically produce models with good semantic tagging results.

Our focus in this paper is on supervised learning models. Deep learning models (or deep models in short) have become popular methods for semantic tagging today. One reason why deep models are popular for semantic tagging is that they are often more capable of learning complicated functions than other kinds of models. Another reason is that the superiority of deep models has been reported by many publications. For example, deep models achieve good prediction quality that is close to the human prediction on GLUE SST-2 sentiment classification task [50, 13]. Some recent studies [35, 34, 49, 15, 51, 37, 33, 18, 46, 52] made comparisons between deep models and simple models (i.e., machine models that do not leverage deep learning) to understand whether deep models are always superior to simple models. They conducted comparisons on various tagging tasks such as suggestion mining [36] or humor detection [40]. Their results reveal marginal or sometimes no improvements of deep models over simple models. It is therefore natural to ask whether deep models are better than simple models when developing solutions for semantic tagging.

Semantic tagging forms the core of many tasks including sentiment classification, suggestion mining, and humor detection. Existing studies, however, compare deep and simple models only on individual tasks. Furthermore, they do not provide insights on how dataset characteristics affect the performances of different models. Consequently, it is hard to generalize their model selection criteria to new tasks or new datasets for the same task. Hence, given a new dataset, it is still unclear whether selecting a deep model will bring the best tagging performance.

In this paper, we embark on a systematic study to understand the performance tradeoffs of deep models vs. simple models for semantic tagging. Towards this goal, we selected 3 representative deep models: CNN, LSTM, and BERT and 2 representative sim-

ple models: LR and SVM. CNN [23] and LSTM [17] are well-known methods that have been widely used in both the academic and industry communities and more recently, BERT [10]. To make a meaningful comparison and systematic study, we collected 21 real datasets that are frequently used in semantic tagging. These datasets exhibit several prominent data characteristics, including (1) a variable number of labels (thousands to millions); (2) a wide range of tag-conveying label ratio (1.6% to 71.4%, ratio $<$ 25% considered as imbalance in this paper); (3) different label cleanliness (clean and dirty labels).

We evaluate the quality of semantic tagging on five selected models on 21 datasets and we obtain a rather surprising finding. We find that deep models and simple models are complementary to each other on the task of semantic tagging. Specifically, deep models perform significantly better on smaller datasets, while simple models can be trained more efficiently on larger datasets and achieve similar semantic tagging quality. Therefore, one should select deep models or simple models for semantic tagging based on the actual dataset characteristics and requirements on efficiency.

Based on our findings, we develop a comprehensive heat map to guide practitioners on selecting the appropriate model for the desired semantic tagging performance for their datasets. This heat map shows the characteristics of the datasets and their quality score of semantic tagging with different tagging models. By using this heat map, practitioners can estimate the semantic tagging quality gain while adopting different deep or simple models. At the same time, they can also try to improve the dataset characteristics to improve the quality of semantic tagging.

Contributions. We systematically evaluate deep models and simple models for the task of semantic tagging. Our key contributions are as follows. (1) We surveyed a number of applications to motivate our study. We selected three representative deep models and two simple models that are widely used to develop these applications. We collected 21 datasets of varying characteristics for a comprehensive study. (2) We conducted extensive evaluations to obtain performance of semantic tagging of the five selected models on all the datasets. We found deep models do not necessarily perform better than simple models on large datasets. (3) We evaluated the effects of dataset characteristics on the quality of semantic tagging. We found the training size, label ratio, and label cleanliness impact the quality of semantic tagging. (4) We generated a comprehensive heat map that can guide practitioners to decide whether they should adopt deep models or simple models and anticipate the performance of semantic tagging for their datasets. To facilitate future research, we will release our collection of datasets, models, and implementations at https://github.com/rit-git/tagging.

Outline. We survey a number applications in Section 2. We discuss the designs of selected deep models and simple models in Section 3. We introduce the collected datasets and their characteristics in Section 4. We perform experimental evaluations and comparisons in Section 5. We analyze the effect of dataset characteristics and present key findings in Section 6. We conclude our study in Section 7.

2. LATEST APPLICATIONS OF SEMANTIC TAGGING

Semantic tagging has broad applications in the field. In this section, we focus on introducing the latest ones. These novel applications come from publications and industrial practices in recent three years. We investigated related literature and link them to five

Table 1: TripAdvisor city guide tips

City	Tips
Washington DC	1. 20% tip is customary for most services 2. Avoid walking through parks at night
Hong Kong	1. Many 'local' restaurants offer menus in English. Ask if they have one before you sit down. 2. Grab an Octopus Card to store money and use on buses, trains and in convenience stores (you have to pay deposit). It will save you a lot of time queuing or fishing for change.

new tags, i.e. Tip, Product Description, Humor, Spoiler, and Argument. These new tags and applications can contribute more ideas to practitioners to leverage semantic tagging on their datasets.

2.1 Tip

Experience-sharing websites such as TripAdvisor and Yelp encourage users to write short yet useful tips such as those in Table 1 and Table 2. This motivates the tip tagging task, which automatically identifies tip-expressing sentences from reviews instead of mandatorily asking users to do so. In comparison to a full review, which is sometimes lengthy, a tip is much shorter yet still reveals the most prominent features of targets.

A tip usually contains suggestive information that enables future customers to improve their consumption experience. For example, a tip may recommend paying 20% tips for services in Washington DC or ask for English menus in Hong Kong as in Table 1. Guy et al. [15] studied tagging practical and short texts from reviews. The authors noticed that readers might feel overwhelmed by reading massive reviews and tend to miss key points. Also, a lengthy review is not friendly to cell phones as the screens have limited space. Wang et al. [51] explored how to highlight tip sentences in software review to help developers pinpoint potential improvements of the software. The developers feel easier to identify emerging issues or prominent problems by reading tips. Weber et al. [52] analyzed queries sent to Yahoo Search and found a large fraction of queries containing how-to intent, such as "how to zest a lime without zester". They tagged tip answers from Yahoo!Answer databases and showed the tip answer at the front of the screen when receiving a how-to query from the user. The conciseness and practicality of a tip help users obtain answers quickly. Zhu et al. [59] observed that a practical tip tends to mention the speciality of a service, i.e. aspects frequently discussed in reviews of this service but less frequently in reviews of other services.

A number of studies focus on tips that give suggestions explicitly. Sapna et al. [35] asked human annotators to label explicit customer-to-customer suggestions that meet two conditions: (1) explicitly state the intention of giving a recommendation; (2) have the potential to benefit customers. For example, a suggestion sentence to be labeled may look like 'Try the cup cakes at the bakery next door'. An opposite example may look like "the cup cakes from the bakery next door are delicious". Since suggestion-expressing sentences are valuable, tagging them from massive reviews becomes an essential task for SEMEVAL 2019 competitions [36]. To perform this task, competitors built classifiers to predict whether a sentence expresses a suggestion or not. The task provided 9092 and 808 labeled sentences in software and hotel domain correspondingly. The best performing algorithm [28] adopted the recent pretrained BERT [10] model that can work decently with a small number of training data.

Table 2: Yelp tips

Business	Tips
Restaurant	1. There aren't many sushi choices in Mountain View, so take what you can get. 2. Free Parking in the lot across the street and around the residential areas.
Coffee shop	1. only place in Bay Area I've been able to find soft boiled eggs 2. Delightful cafe with solid pastries. Drinks left more to be desired and service is a bit slow, so make sure you have ample time!.

2.2 Product Description

E-commerce websites such as Amazon and eBay collect a lot of customer reviews in recent years. Among these customer reviews, product description is one of the most informative sentences. Product description from customer reviews often contains supplementary information that is missing in official product descriptions. These customer product descriptions are attractive to people due to multiple reasons. First, they come from independent users who have no personal connections with merchants. Second, they summarize crucial facets concerning actual user experience. Third, they introduce exceptions to avoid unnecessary orders when the product is not applicable. For example, a buyer posted actual experience after using a camera for several weeks. He described the actual performance of the camera, such as the photo quality, the ergonomics design, and the battery duration. He might also describe the returning policy and experience if he did not like the camera. Therefore, other customers can determine whether to buy a product according to these customer reviews.

Tagging product description from reviews serves critical purposes. A study [37] about eBay reported that official descriptions were always lacking for new items (e.g., fashion products). For old items, details were also frequently missing from official descriptions. To solve this problem, the authors tagged user descriptions from customer reviews and conducted a user study to see whether customer descriptions could provide more information. The user study showed that people believed customer descriptions were more informative and objective than sentimental sentences such as "It was the best socks ever". One example of how authors tagging customer descriptions is shown in Example 1. Some other studies also showed how to tag customer descriptions and apply them to industrial products. Rakesh et al. [43] proposed to tag and organize customer descriptions according to different aspects of a product. They further presented a review visualizing tool empowered by product-describing sentences based on their proposal. Mitcheltree et al. [32] distilled aspect-describing sentences of a user and used these sentences to understand the tastes of this user. Zhang et al. [57] detected defects of a product from customer reviews and provided instructions for customers when using the corresponding product. These studies have substantially simplified the purchase decisions for customers.

User-written product descriptions help merchants and readers better understand sentimental descriptions. This is especially important as pure sentiment words only provide none objective information. Furthermore, merchants expect users to describe products objectively instead of expressing sentiments that might expel potential buyers. On the other hand, customer readers also look for objective evidence to avoid being misled by sentiments. Therefore, customer descriptions are complementary to sentimental comments and necessary for customers to choose suitable products.

EXAMPLE 1. *Perfect thickness for shoes or boots. Extra padding at toes. The quality is excellent. Easy to handle and very comfortable. No shrinkage in washer or dryer.*

2.3 Humor

Humor has the power to bring people happiness. YELP [6] started exploring humor in customer reviews and offered user interfaces to label whether a review is funny or not. Example 2 and 3 show two examples of commenting "clean room" and "delicious buffet". Although informative online reviews can help us make decisions, reading, and digesting these reviews are stressful and unpleasant. Different from others, a humorous review can bring happiness to the readers, even it can not provide more information. Therefore, humorous reviews can attract more attention from users for those apps who prioritize them than other reviews.

Tagging humorous text from a corpus attracts a broader discussion recently. Yang et al. [53] studied humor extraction that classifies whether a given text contains humor or not. They identified four types of features that could better classify humor. Morales and Zhai [33] tried to tag humor from merchant reviews as the YELP Challenge released a large annotated corpus. Each of the 6,685,900 reviews contains the number of votes from readers who thought the review was funny. Cattle and Ma [5] emphasized that a humor sentence expresses semantics that goes beyond human expectation. Thus humor-expressing sentences tend to contain words that are semantically different. Blinov et al. [3] prepared a dataset of funny dialogues in Russian. They calculated the length distribution of both jokes and non-jokes and found that a text with more than 100 characters is likely to be a joke than non-joke. Yu et al. [54] studied how to generate pun with neural networks. Due to the prevalence of humor, SEMEVAL 2017 held two tasks on humor detection [31, 40]. These studies reveal the essence of humor and also the unique features of humorous comments among all types of sentences.

EXAMPLE 2. *Restaurant so clean....you can eat off the floor!!!!*

EXAMPLE 3. *I usually am NOT a fan of buffets.. but this place really puts other buffets to SHAME!*

2.4 Argument

Arguments provide evidence to support or oppose an opinion and are useful for explanations. Compared with pure sentimental comments such as "the room is uncomfortable", argument sentences usually contain factual details such as "the room has a smell of mold". Since arguments are generally different opinions towards the same proposition, they often bring much information from wide perspectives. Example 4 and 5 show how argument and non-argument sentences pointing to the proposition "nuclear energy is good" from questia [42]. These two examples demonstrate that argument sentences provide more explanatory information towards a proposition.

Argument mining studied automatic tagging of argument expressing sentences that had caused extensive discussion in review mining. Hua et al. [18] started exploring argument tagging in the domain of paper reviews, as they noticed paper reviewing takes approximately 63.4 million hours in 2015. They obtained public reviews from ICLR and extracted argument sentences from these reviews. Then they classified the argument sentences into five classes (i.e. request, evaluation, reference, quote, or fact) to better understand peer comments. Stab et al. [46] studied argument mining for general text. They collected datasets from Questia and annotated arguments for eight controversial-topics such as abortion and marijuana. Kim and Hovy [22] started mining reasons depicting the pros and cons of the targeted service or product. Poddar et al.

[38] applied argument mining to customer reviews to tag supporting opinions for a given aspect. These studies provide us more knowledge of argument sentences and show us their potential applications in actual products.

EXAMPLE 4. *The entire nuclear fuel chain generates lots of long-lasting radioactive waste.*

EXAMPLE 5. *Can truly green, renewable sources of energy replace nuclear power?*

2.5 Spoiler

Spoiler commonly exists in the reviews of media works such as TV series. In such reviews, audiences express their evaluations and ideas towards media works, while they inevitably quote specific plots to support their ideas. These spoilers might ruin the expectation and enjoyment of audiences who have not watched the media works. Therefore, a spoiler alert is necessary to warn readers before they browse the reviews. For example, the movie encyclopedia website IMDB mandatorily requires reviewers to add an alert message to the title if they are going to include spoilers in the comments. Online wiki Lostpedia also banned spoilers since mid-2008. Due to the broad necessity of spoiler alerts, automatic detection becomes more desirable so that reviewers can avoid writing spoilers, and readers can avoid reading spoilers.

Spoiler detection is a field with little exploration, mainly due to the shortage of datasets. There are only two open-source datasets for spoiler detection. One is TV Tropes dataset released by Boyd-Graber [4]. This dataset contains spoiler sentences that disclose important TV show plots, such as the ending of main characters. An example of TV Tropes is shown in Example 6. The other one is a large dataset consisting of millions of book spoiler sentences released by Wan et al. [49]. These authors found that spoiler sentences tend to appear in the later part of a review and book spoilers are much variable in words correspondingly to the book-specific contents, such as characters' names. Due to the ubiquitousness of spoilers and the necessities of automatic detectors, this paper received a number of media reports. An example of book spoilers is shown at Example 7.

EXAMPLE 6. *None of the Harmons survive.*

EXAMPLE 7. *This book had all the potential to be great: a political thriller with tons of twists including, but not limited to, killing off the main character in the middle of the book .*

3. REPRESENTATIVE MODELS

In the previous section, we saw a sentence can have five types of tags. For each type, a binary classifier can be trained using example data. We first formally define the semantic tagging problem and describe the common training pipeline. We then describe the representative sentence classification models for semantic tagging.

3.1 The Semantic Tagging Problem

The semantic tagging problem is generally solved by a binary text classification solution. The solution enables people to classify a text into one of the two classes: {*desired, not-desired*}. Based on the classification results, the tagged sentences are essentially those classified as "*desired*".

The sentence classification problem is widely studied as text classification [24, 1] in the machine learning and natural language processing community. To solve the supervised sentence classification problem, people generally perform experiments at three steps: (1).

label data preparation; (2). input sentences representation; and (3). model selection. The quality of each step can significantly affect the accuracy of the classification.

Label data preparation. The first step is label preparation. A label instance is in the format of $(text, label)$, where $text$ is the raw sentence, and $label$ is either 1 (positive training example) or 0 (negative example). After we have collected a number of labels, we apportion the data into training and test sets, with a pre-specified split (e.g., 80-20). The purpose of splitting data is to mimic actual prediction, where the model training is performed on the training set but the model evaluation is conducted on the testing set.

Input sentence representation. The second step is pre-processing that raw sentences are converted into numeric representations by classification models. At this step, a sentence is tokenized into words. The purpose of tokenization is to represent each word using a numeric word vector. By aggregating word vectors, we can obtain the final numeric representation of the sentence. There are two commonly used sentence representation methods: Bag-of-words (BoW) and Word embeddings. The choice of a method heavily depends on the choice of the classification model.

Model selection. Classification models include two sub-categories: statistical models and neural network models. Compared with neural network models, statistical models are usually simpler (smaller in size) and less computationally intensive. However, statistical models require more domain knowledge in constructing a set of essential features for each specific task. In contrast, neural network models are larger in size (more parameters) and more computationally intensive (usually trained on hardware accelerators like GPUs), but they do not require feature engineering and can learn how to represent the text directly from training data. Due to this trade-off, people often select different models for different scenarios.

However, we are witnessing a rise in the popularity of neural network models, such that they are considered best performing models for all semantic tagging tasks. We revisit this popular belief and compare an array of statistical models (Logistic Regression (LR) and Support Vector Machine (SVM)) and deep models (Convolutional Neural Network (CNN) [23], Long Short-Term Memory (LSTM) [17], Bidirectional Encoder Representations from Transformers (BERT) [10]).

3.2 Simple Models (LR/SVM)

We use Bag-of-words (BoW) as the input representations for simple models. First, BoW splits a text into tokens, each of which can be a unigram (a single word) or a bigram (two consecutive words). In our experiments, we found a combination of unigram and bigram yields the best tagging quality. Second, BoW calculates the number of distinct tokens as vocabulary size (denoted as d), so that each token has a unique position in a d-dimensional vector. Third, BoW represents a text as a d-dimensional vector, where the i-th position records the count of the corresponding token in the text. So there will be many 0 in the vector if the vocabulary size d is large.

BoW also uses IDF (inverse document frequency) to weigh each of the d tokens. IDF assumes that a token is more important if it appears in less number of sentences, and the formula of IDF is $idf(t) = log[n/df(t)] + 1$, where t is the corresponding token, n is the total number of sentences, and $df(t)$ is the number of sentences that contain t. For example, "My advanced geometry class is full of squares" is a pun-containing humorous sentence, in which the word "my" is less informative than the word "geometry" and "squares".

Logistic Regression (LR). A classical simple model is LR [12], which is derived from linear regression and uses Sigmoid [12] function to scale the output into a real number between 0 and 1. This real number indicates the probability of an instance belonging to a class. LR is widely used in production to analyse large-scale data [16, 2]. It has the advantages of low computational overhead and high parallelism [19, 55].

Support Vector Machine (SVM). Another popular simple model is SVM [47], which separates two classes by their border points. We therefore use linear SVM, which adopts a straight line, to separate different classes. Linear SVM is known to be empirically effective in high-dimensional space, where data become sparse and tend to be linearly separable [47, 21, 41, 20]. Besides, linear SVM scales to large number of labels better than its variants of using non-linear kernels in training time [45, 7].

3.3 Deep Models (CNN/LSTM/BERT)

Convolutional Neural Network (CNN). CNN [23] has traditionally been used for image classification but is now used for several NLP tasks. CNN tokenizes a text into unigram words. Each word is represented with a pre-trained k-dimensional vector. Two words will have similar word vectors if they are semantically similar (e.g., synonyms). The text is represented as a $m \times k$ matrix, where m is a predefined maximum sequence length. In this paper, we also name the $m \times k$ matrix as feature matrix. CNN consists of multiple convolutional layers and pooling layers. The convolutional layers convert the consecutive elements (e.g., bi-grams like "please add") of the input matrix into a sequence of feature vectors with a sliding window. Then the pooling layers aggregate the sequence into a shortened one via a max aggregate function. The convolutional and pooling layers are usually stacked to obtain a single vector that represents the sentence meaning. Intuitively, when CNN is applied on text sequences, the neural network first constructs low-level features from local information like bi-grams and tri-grams, and then constructs higher-level features (e.g., whether a span contains related information) from the low-level ones.

Long Short-Term Memory (LSTM). LSTM [17] is based on recurrent neural network (RNN) [14]. It has been shown to be effective for several tasks including classification, tagging, and translation. LSTM uses the same input representation as CNN, i.e. a matrix of m word vectors. However, unlike CNN, LSTM sequentially (left to right) processes the text over time and keeps its *hidden state* through time. The hidden state can capture any meaningful features that appeared in the prefix of the text up to the current timestamp. This enables LSTM to capture arbitrary long-term dependencies. Compared with vanilla RNN, LSTM (and its variants like GRU [9]) partially solves the gradient exploding and vanishing problem that results in improved model performances with its specially designed components (i.e. input/output/forget gates).

Bidirectional Encoder Representations from Transformers (BERT). BERT [10] is an award-winning state-of-the-art model. Similar to CNN and LSTM, BERT also uses a matrix of word vectors to represent a text. BERT applies attention technique to represent a text with weighted word vectors, such that relevant tokens have higher weights than irrelevant ones (e.g. "great" and "delicious" as relevant tokens and "that" as irrelevant token in YELP polarity classification). BERT derives its performance from language representation pre-trained on a large corpus, Wikipedia. So BERT can learn a default vector and the weight of a word in the

Table 3: Statistics of 21 Dataset. We obtained 19 datasets in 5 different application domains. We further created two additional large datasets from FUNNY and BOOK by balancing their positive and negative labels. Among all 21 datasets, 6 of them are large datasets with more than 100K records (highlighted in gray). The labels of FUNNY, BOOK, FUNNY*, and BOOK* are dirty due to the existence of missing annotations.

Dataset	Application	#Record	%Positive	Vocabulary
Original Datasets				
SUGG [36]	Tip	9,092	26.2%	10K
HOTEL [35]	Tip	7,534	5.4%	7K
SENT [51]	Tip	11,379	9.8%	8K
PARA [51]	Tip	6,566	16.8%	8K
FUNNY [33]	Humor	4.75M	2.5%	571K
HOMO [31]	Humor	2,250	71.4%	5K
HETER [31]	Humor	1,780	71.4%	5K
TV [4]	Spoiler	13,447	52.5%	20K
BOOK [49]	Spoiler	17.67M	3.2%	373K
EVAL [18]	Argument	10,386	38.3%	8K
REQ [18]	Argument	10,386	18.4%	8K
FACT [18]	Argument	10,386	36.5%	8K
REF [18]	Argument	10,386	2.0%	8K
QUOTE [18]	Argument	10,386	1.6%	8K
ARGUE [46]	Argument	23,450	43.7%	21K
SUPPORT [46]	Argument	23,450	19.4%	21K
AGAINST [46]	Argument	23,450	24.3%	21K
AMAZON [58]	Sentiment	3.6M	50.0%	1M
YELP [58]	Sentiment	560,000	50.0%	232K
Additional Datasets				
FUNNY*	Humor	244,428	50.0%	171K
BOOK*	Spoiler	1.14M	50.0%	112K

Wikipedia corpus. When applying BERT to domain-specific data, these word vectors and weights are optimized according to labels, such that domain-specific knowledge can be incorporated.

4. DATASETS

We curate a collection of 21 textual datasets to evaluate the quality of different models on the semantic tagging task. To the best of our knowledge, our datasets have the best coverage of the real-world datasets for semantic tagging. We focus on three characteristics of the datasets that are crucial to a model's performance: size, label ratio, and label cleanliness. Datasets in our collection have 1,700 to 17,000,000 examples. The ratio of positive instances in our datasets ranges from 1.6% to 71.4%. 4 of the 21 datasets use labels generated based on incomplete metadata, while the rest use either human-annotated labels or labels which are derived from complete metadata.

Source. We collected our datasets from various semantic tagging applications that are surveyed in Section 2. We describe the collection process for each dataset below.

- **Tip:** We use the SUGG dataset from the 9th task of SEMEVAL competition 2019 [36]. The labels are based on whether a user comment can help improve windows software. The HOTEL [35] dataset is based on hotel reviews. The labels are derived based on whether a review sentence provides suggestions for future customers. The SENT [51] and PARA [51] datasets also contain tip sentences (customers' recommendations for software updates) and non-tip sentences (e.g., customers' experience of using the software).
- **Humor:** We use the FUNNY [33] dataset from the YELP Dataset Challenge [6], in which a review received votes from readers if they think this review is funny. Reviews with more than 5 votes are annotated as positive and reviews with 0 votes are annotated as negative [33]. We use the HOMO and HETER datasets from

the 7th task of SEMEVAL competition 2019 [31]. A sentence is labeled positive or negative depending on the occurrence of pun words.

- **Spoiler:** The TV [4] and BOOK [49] datasets are obtained from TV show comments and book comments, respectively. Each sentence is labeled depending on whether the sentence contains a spoiler. Having more than 17 million examples, BOOK is the largest among all the datasets.
- **Argument:** We have 8 datasets for Argument application namely, EVAL, REQ, FACT, REF, QUOTE, ARGUMENT, SUPPORT, and AGAINST. Each dataset has been derived from two multi-class tagging tasks. EVAL, REQ, FACT, REF, and QUOTE [18] include examples of different types of sentences that make up a paper review. ARGUMENT, SUPPORT, and AGAINST [46] are obtained from online discussions of controversial topics and contain sentences corresponding to argument opinion, supportive opinion, and opposed opinion, respectively.
- **Sentiment:** AMAZON [58] and YELP [58] are two Sentiment datasets that are obtained from AMAZON and YELP reviews, respectively. A review is annotated as positive if the writer gives a rating of 3 or 4 and as negative if the rating is 1 or 2.

FUNNY* and BOOK* are 2 additional large datasets that we obtained from FUNNY and BOOK by balancing the positive and negative labels. We randomly drop a number of negative labels to make the label ratio balanced. Overall, the datasets we collected offer a good representation of real-world datasets and enable us to understand the strengths and limitations of different models.

Dataset preparation. Since dataset preparation can directly influence tagging performance and model selection, it is a critical component of the tagging pipeline. Our datasets come from different papers and public competitions, and therefore, must be transformed into the same format. A data record is in the format of (text, label), with label 1 representing the text we wish to tag and 0 otherwise. We calculate crucial information about these datasets in Table 3, such as the application that a dataset belongs to, the number of records, and the percentage of positive labels.

Dataset Characteristics. We next describe the characteristics of the various datasets we collected/generated.
- **Size:** The scales of the datasets skew towards two ends. 15 out of 21 datasets have a small number of records at the scale of 1,000 to 99,999 (small datasets), while the remaining 6 datasets contain 100,000 to 20,000,000 records (large datasets). Small datasets are more common than large datasets in actual cases. This is because every record of a dataset requires human annotation. The high price of recruiting human annotators limits the scale of datasets. However, accompanying the emergence of experience-sharing websites such as Amazon and YELP, more and more large datasets appear. Following this, numerous customers share their opinions on these websites through sentence annotation. For example, YELP users share their opinions on YELP about whether a review is funny [6] or not. Given the polarization of real-word datasets in scale, a practical tagging model should be able to tag both small and large datasets accurately.
- **Label Ratio:** Label imbalance is a common phenomenon in our datasets. We observed 14 out of 21 datasets having fewer positive labels than negative labels. More specifically, 10 of the 14 datasets exhibit a ratio of positive labels smaller than 25% (considered as imbalance). For the remaining 7 datasets, 5 datasets exhibit a balanced ratio of around 50% and 2 datasets exhibit a ratio of positive labels larger than 70%. A higher percentage

Table 4: Dataset taxonomy

Category	Datasets
Small-L	HOTEL, SENT, PARA, REQ, REF, QUOTE, SUPPORT, AGAINST
Small-H	SUGG, HOMO, HETER, TV, EVAL, FACT, ARGUE
Large-L	FUNNY, BOOK
Large-H	AMAZON, YELP, FUNNY*, BOOK*

of positive labels is favorable due to two main reasons. First, more positive instances can increase the size of effective training data, leading to better training quality. Second, more positive instances can minimize the misleading effects of negative instances, increasing the accuracy of tagging results.
- **Cleanliness:** There are two common ways to obtain data labels: rule generation and human annotation. Relying on rules can potentially introduce dirty labels due to missing annotations. For example, relying on number of votes to generate label FUNNY can introduce dirty labels for new records which do not have enough votes. Similarly, relying on spoiler alters to generate label BOOK can be dirty if writers do not leave any alert on their reviews. Consequently, we consider FUNNY, BOOK, and their derivatives (FUNNY* and BOOK*), based on rule generation, as dirty datasets. The rest of the datasets, which are based on human annotation, are categorized as clean datasets.

5. EMPIRICAL EVALUATION

In this section, we compare the tagging performance of representative models, including LR, SVM, CNN, LSTM, and BERT on the datasets we collected. Experiments show that deep models do not consistently outperform the simple models across all datasets. On large datasets, in particular, deep models obtain similar or worse performance, yet they take significantly more training time.

The section is organized as follows. We introduce experimental settings in Section 5.1. We report detailed performance of different models in Section 5.2. We focus on analyzing BERT in Section 5.3 since BERT consistently outperforms other deep models.

5.1 Experimental Settings

Dataset taxonomy. We had introduced the 21 datasets in Section 4. We categorize the datasets into four groups based on size and label ratio (whether the number of records is larger than 100,000 and the percentage of positive labels is larger than 25%). The dataset taxonomy is shown in Table 4. It is based on 4 categories namely, small size low percentage (Small-L), small size high percentage (Small-H), large size low percentage (Large-L), and large size high percentage (Large-H.). The taxonomy helps understand models' performance in terms of characteristics of the datasets.

Dataset Preparation. We split each dataset into a train set and a test set. Train set contains 80% records while test set contains the remaining 20% records. We do not split SUGG, HOMO, and HETER, since each of them already contains a separate test set for SEMEVAL competitions [36, 31]. The train set ratio of SUGG, HOMO, and HETER is 93%, 80%, and 80%, respectively.

Computing resource. We conduct experiments on an AMAZON EC2 p3.8xlarge GPU server. Our instance is equipped with 4 Tesla V100 GPUs. Each GPU has 16GB memory, 640 Tensor cores, and 5120 CUDA cores. The server is running on Linux Ubuntu 16.04. The monetary cost is $3 US dollars per hour of using the GPUs.

Hyper-parameters. We tune each model to the best performance according to common practices. We found using a combination of unigram and bigram in Bag-of-words representation yields the best tagging quality for LR and SVM. We adopt the default setting for BERT [10]. We set the batch size to 32, the max sequence length to 128, and the number of epochs to 3. For CNN and LSTM, we use the same batch size as BERT but set the max sequence length to 256 and the number of epochs to 10. We do not observe clear performance improvement when using larger batch size, longer max sequence length, and more epochs for CNN and LSTM.

Evaluation Metric. We measure the quality as well as the training cost of tagging positive instances. We use F1 and training time accordingly. F1 is an averaged quality indicator and defined as $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$, where precision and recall are standard information retrieval metrics. For example, if we assume that there are 10 positive instances and an algorithm tags 8 positive instances with 6 are correct, the precision is $6/8 = 0.75$ and recall is $6/10 = 0.6$. In this case, the F1 equals to $2 * 0.75 * 0.6/(0.75 + 0.6) = 0.66$.

Macro- and Micro-average F1 When evaluating a set of datasets, we report macro- and micro-average F1 to compare the overall performance of simple and deep models. Macro F1 is the average F1 score, for which it is insensitive to the sizes of datasets. Given that the 21 datasets are of various sizes, we also calculate micro-average F1, the sum of weighted F1s, as a complement to macro-average F1. Specifically, the weight of a dataset is the number of records of this dataset divided by total number of records of all datasets. Therefore, a larger dataset will have a higher weight than a smaller dataset.

5.2 Result Analysis

We first describe the performance of each individual model. Next, we compare the performance of different models on different dataset categories. Lastly, we discuss their performance and training time trade-offs.

5.2.1 Performance of individual models

Table 5 shows the macro- and micro- average F1 score of each individual model on datasets grouped according to the taxonomy. Figure 1 and Figure 2 show the F1 scores on each dataset. Figure 1 shows the F1 scores of individual model on the small and large datasets with high positive label ratio ($\geqslant 25\%$). On the other hand, Figure 2 shows the F1 scores on the small and large datasets with low positive label ratio (i.e. imbalance).

LR and SVM. LR and SVM achieve very similar F1 (difference $< \pm 0.03$) on all the datasets except FUNNY. Both LR and SVM achieve the lowest F1 (0.10 and 0.10) on QUOTE and the highest F1 (0.94 and 0.96) on YELP. Compared with QUOTE, YELP has a larger size (560,000 vs. 10,000) and a higher label ratio (0.5 vs. 0.02), indicating the size or the label ratio of a dataset has significant effect on the performance of simple models. As shown by Figure 1 and Figure 2, LR achieves an average F1 of 0.79 on the datasets with the positive label ratio $\geqslant 25\%$ and 0.46 on the positive label ratio $< 25\%$. SVM shows a similar behavior on datasets with different ratios of positive labels. This suggests that the tagging quality of simple models is significantly affected by the ratio of positive labels.

CNN and LSTM. The F1 score of CNN ranges from 0.08 to 0.94, while the F1 score of LSTM ranges from 0.11 to 0.93. Interestingly, these scores are lower than F1 scores of LR and SVM, indicating

Table 5: The (macro-/micro-) average F1 of LR, SVM, CNN, LSTM, and BERT in dataset categories

	LR	SVM	CNN	LSTM	BERT
Large-H	0.85/0.77	0.85/0.76	0.80/0.72	0.80/0.72	0.87/0.79
Small-H	0.77/0.73	0.76/0.72	0.75/0.70	0.75/0.71	0.85/0.82
Small-L	0.52/0.51	0.52/0.51	0.49/0.47	0.51/0.49	0.68/0.66
Large-L	0.23/0.20	0.27/0.20	0.07/0.06	0.12/0.11	0.24/0.19

that deep models do not always outperform simple models. Our finding contradicts the popular belief that deep models are the best choice for semantic tagging task. Furthermore, we find that CNN and LSTM are also sensitive to positive label ratio. The average F1 of CNN is 0.77 on datasets with a ratio $\geqslant 25\%$, and 0.39 on datasets with a ratio $< 25\%$.

BERT. BERT gains wide-spread popularity in recent years and is considered a de-facto model for semantic tagging tasks. Not surprisingly, it does achieve highest F1 scores on most of the datasets (19 of 21). This is perhaps because it is pre-trained on a large corpus and hence is optimized for large datasets. However, our experiments reveal that BERT does not show apparent advantages over simple models on large datasets. The average F1 scores of BERT, LR, and SVM on all the 6 large datasets are 0.66, 0.64, 0.66, respectively. In fact, on 2 of the large datasets which are also imbalanced, BERT performs worse than simple models. These results suggest that deep models may not always perform better, especially on large or imbalanced datasets.

Other industrial models. We also evaluate the performance of more simple and deep models that are extensively used in industry. The newly investigated simple models include Naive Bayes [30] and XGBoost [8] (a ensemble/boosting model). The newly investigated deep models include ALBERT [25] and ROBERTA [29], which are obtained from Huggingface transformers [48]. We report average F1 scores of Naive Bayes and XGBoost in Figure 3a. We also include the results of LR and SVM in the same figure for comparison. The average F1 of LR, SVM, Naive Bayes, and XGBoost are overall similar, so we use LR and SVM as representative simple models in our paper. We report average F1 of ALBERT and ROBERTA in Figure 3b. We also include F1 of BERT in the same figure for comparison. Overall, BERT shows slightly better F1 than ALBERT and ROBERTA, so we take BERT as the representative attention-based deep model in our paper. We report the results of F1 scores for individual datasets in Appendix of the technical report [26].

5.2.2 Comparison on different dataset categories

Next, we analyze the performance of the models by dataset categories. Table 5 shows the macro- and micro- average F1 scores of different models.

Large and high datasets. Overall, all models perform the best on datasets that are large in size and high in the ratio of positive labels. This is reflected in higher F1 of every model on the Large-H dataset category than on other categories.

Small and high datasets. The average F1 scores of different models on the Small-H datasets are slightly lower than F1 scores on Large-H datasets. On the other hand, average F1 scores on Small-L datasets are much lower. This suggests that the size of a dataset
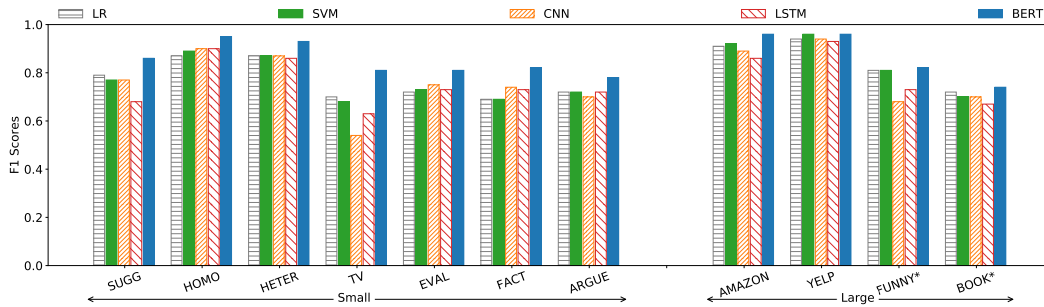
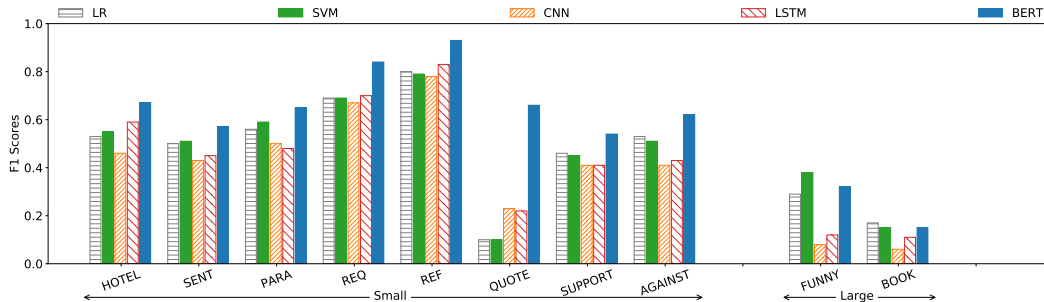**Figure 1:** F1 on Small-H and Large-H datasets with $\geqslant 25\%$ positive labels

Figure 2: F1 on Small-L and Large-L datasets with $< 25\%$ positive labels
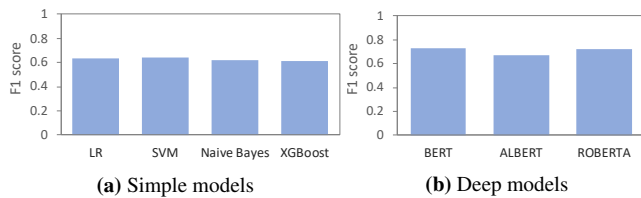

Figure 3: The (macro) average F1 of industrial models on all 21 datasets

does not affect tagging quality significantly when the ratio of positive label is high.

Small and low datasets. The average F1 scores on Small-L datasets are significantly lower than on Small-H datasets. This indicates that a low ratio of positive labels (i.e. imbalance) can negatively affect the tagging quality of models.

Large and low datasets. Intuitively, large number of labels offer more training examples which should lead to high F1. However, we find that F1 scores on Large-L datasets do not follow this intuition. The average F1 scores of different models on Large-L datasets are the lowest among the different dataset categories. There are two reasons for low F1 scores. First, ratios of positive labels of the two datasets are significantly lower compared with other datasets. Second, labels of the two datasets are dirtier than other datasets, as we mention in Section 4. Interestingly, simple models achieve even better F1s than deep models on the two datasets that may due to poor label cleanliness and/or severe imbalance. This suggests that model selection should be done more carefully for Large-L datasets with many dirty labels.

Multi-label datasets. We also perform comparison on multi-label datasets. We find two datasets, BIO and DEF, from task 6 of the SEMEVAL2020 competition [44]. BIO is a Named Entity Recognition dataset while DEF is a Knowledge Graph Extraction dataset. BIO contains around 470, 000 labels so the dataset is large. Each label is associated with a word token, and is either B, I or O. The best performing simple and deep models show similar F1s (same regarding labels B and O, 0.02 difference regarding label I). The results indicate that simple models can achieve similar performance as deep models when the number of labels is sufficiently large. DEF contains around 18, 000 labels so the dataset is small. Each label is associated with a sentence, and is either T or F. The best performing deep model outperforms the best simple model on F1 (0.14 gap regarding label T and 0.04 improvement regarding label F). The results indicate that deep models achieve better performance than simple models when the number of labels is small. We present the detailed results and discussions in Appendix of the technical report [26].
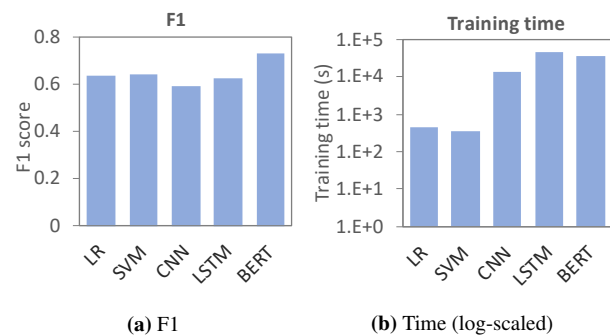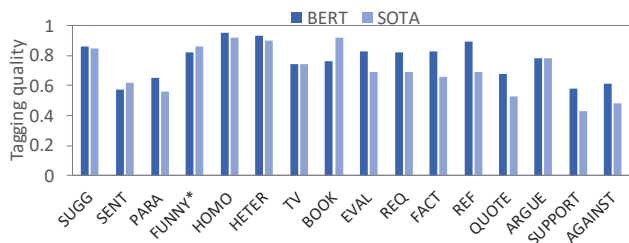

Figure 4: The (macro) average F1 score and average training time (log-scaled) on all datasets

Figure 5: BERT compares with SOTA (the state-of-the-art). We follow the quality metric in SOTA publications (F1 by default, Accuracy for FUNNY [33], TV [49], and AUC for BOOK [49].)

5.2.3 Performance/Training time trade-off

Figure 4a shows the (macro) average F1 scores of different models across all the datasets. We find that not all deep models outperform simple models. While BERT achieves higher F1 (with a large margin of 0.11) than LR/SVM, CNN and LSTM perform much worse.

We report the training time of the models in Figure 4b. We would like to remind the readers that all the deep models were trained on GPU while the simple models were trained using a CPU. Although some deep models achieve higher F1 than simple models, they also take 30x-130x more time for training. LSTM is the slowest and takes 13 hours on average for training and costs 39 dollars. CNN and BERT are slightly faster but still take a lot of time. In practice, debugging and parameter tuning deep models also take up a lot of time, adding to the overall cost. For example, BERT has a number of tunable hyperparameters, such as maximum sequence length, training batch size, number of training epochs, and learning rate. Each hyper-parameter has multiple options. Even with only 3 options per hyper-parameter, there can be as many as $3^4 = 81$ combinations that will cause significant overhead. On the other hand, simple models take less than 500 seconds using a single CPU. This suggests that simple models can bring more economic benefits to users, especially to those who have no access to GPU's but need to train tagging models on large datasets.

5.3 Analysis of BERT

As shown in Figure 1 and Figure 2, BERT consistently achieves higher F1 scores than other deep models. We, therefore, take a closer look at BERT.

BERT versus domain SOTA. Our results indicate BERT as the most accurate generic model. We, therefore, compare it with the state-of-the-art (SOTA) models, which are considered as golden standards for semantic tagging tasks, on individual datasets. These SOTA results (e.g., 0.85 F1 on SUGG from SEMEVAL 2019 champion [28]) were obtained by leveraging domain-specific knowledge. We follow the experimental settings and metrics given in published materials. Specifically, we compute the F1 score for SUGG [28], SENT [51], PARA [51], HOMO [60], HETER [11], EVAL [18], FACT [18], REF [18], QUOTE [18], ARGUE [46], SUPPORT [46], AGAINST [46], and Accuracy for FUNNY* [33], TV [49] and AUC [49] for BOOK [49]. Following the computing process, we directly take the published SOTA results from related publications and perform comparison between SOTA and BERT.

As shown in Figure 5, BERT achieves comparable or even better results compared with SOTA methods. Based on these findings, practitioners should really consider using BERT as a base model to replace existing solutions or develop new pipelines for semantic tagging. We also notice that BERT does not outperform SOTA in
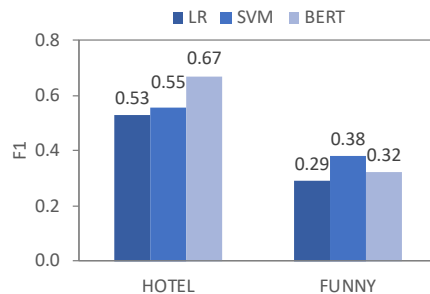
Figure 6: F1s on a small/large dataset HOTEL/FUNNY

Table 6: F1 of LR and SVM without/with pre-trained BERT embeddings (eb.)

Dataset	LR	LR + eb.	SVM	SVM + eb.
HOMO	0.87	0.94	0.89	0.93
HETER	0.87	0.92	0.87	0.91
QUOTE	0.10	0.35	0.10	0.34

SENT, FUNNY*, and BOOK. Regarding SENT [51], we can only obtain partial training set so the F1 scores of BERT and SOTA are not comparable. For FUNNY* [33], the gap is pretty small (0.04). For BOOK [49], domain tagging extensively leverages the names of characters to identify spoilers, while these names are out of the vocabulary of BERT. Except for these extreme cases, BERT is always the best tagging model on most datasets.

BERT on representative datasets. We choose HOTEL, FUNNY as the representative datasets for small, large datasets, respectively, and compare BERT with simple models. As shown in Figure 6, BERT achieves higher F1 than LR (0.14) and SVM (0.12) on HOTEL, confirming that BERT indeed significantly improves tagging F1 in some cases. However, in other cases, BERT does not outperform simple models. For example, BERT performs worse than SVM by 0.06 F1 on FUNNY. Simple models can sometimes outperform BERT while taking significantly less time for training. For example, BERT takes 1.4 days to train FUNNY that contains 4.75 million records. Therefore, if considering both F1 improvement and time consumption as evaluation criteria, deep models are not the best choice for semantic tagging in some cases.

Effect of pre-trained embeddings We conducted experiments to evaluate how much simple models can benefit from pre-trained embeddings. Given that BERT is similar to Word2Vec and Glove that use word vectors, for the convenience, we used BERT that was pre-trained over Wikipedia corpus as the embedder. For each input text, BERT outputs the last-layer [CLS] vector [10] as the featurization vector. We then ran LR and SVM on all featurization vectors and presented F1s of the most representative datasets (i.e. HOMO, HETER, and QUOTE) in Table 6. The results show that LR can achieve better F1s on small datasets with pre-trained embeddings. F1 improvements of HOMO, Heter, and QUOTE are 0.07, 0.05, and 0.25, respectively. Similarly, SVM also achieves better F1s on these three datasets. The F1 improvements of HOMO, HETER, and QUOTE are 0.04, 0.04, and 0.23, respectively. These results suggest that simple models can benefit from pre-trained embeddings. More results and discussions are in Appendix of the technical report [26].

Table 7: Comparison of the best DEEP model and the best SIM-PLE model on different types of datasets

Datasets	Avg F1			Avg train time (s)	
	DEEP	SIMPLE	Δ	DEEP	SIMPLE
Small-L	0.68	0.52	0.16	308	<1
Small-H	0.86	0.78	0.08	324	<1
Large-L	0.24	0.27	-0.03	308,680	3,128
Large-H	0.87	0.85	0.02	38,466	318

6. ANALYSIS BY DATASET CHARACTER-ISTICS

Our empirical evaluation suggests that model performance and, thus, model selection is influenced by dataset characteristics. We, therefore, investigate the effect of size, label ratio, and label cleanliness on model's performance. First, we compare BERT and simple models by dataset types to find more insights in Section 6.1. Next, we investigate the effects of dataset characteristics on model selection in Section 6.2. Lastly, we summarize our findings to help practitioners find the best model for their dataset in Section 6.3.

6.1 Comparison of best models by type

In this section, we summarize our key findings. We only compare the best deep model (i.e. BERT, denoted as DEEP) and the best simple model (denoted as SIMPLE) for simplicity. We report the average F1 and average training time on each type of datasets in Table 7.

Model selection. Our goal is to find out whether DEEP is better than SIMPLE. Our results show: (1). DEEP outperforms SIMPLE by a large margin on small datasets (0.16 on Small-L and 0.08 on Small-H respectively). However, it does not win on large datasets (-0.03 on Large-L and 0.02 on Large-H). (2). Training DEEP is efficient on small datasets (in average 308/324 seconds for Small-L / Small-H), but inefficient on large datasets (in average 86/11 hours for Large-L / Large-H). (3). SIMPLE becomes very competitive on large datasets since it obtains F1 similar to DEEP yet takes much less training time (on average 0.9/0.1 hours for Large-L / Large-H). Therefore, DEEP will be a good choice for small datasets, but a questionable choice for large datasets.

Dataset preference. We observed consistent dataset preference of DEEP and SIMPLE regarding F1. Specifically, (1). both DEEP and SIMPLE achieve the highest F1s on Large-H, the 2nd F1s on Small-H, the 3rd F1s on Small-L, and the smallest F1s on Large-L. The performance of a model subjects to the dataset taxonomy. (2). Both DEEP and SIMPLE favor datasets with high percentages of positive labels. They achieve higher F1 on Small-H than Small-L (0.86 versus 0.68 for DEEP and 0.78 versus 0.52 for SIMPLE). (3). Neither DEEP nor SIMPLE achieves satisfactory performance on Large-L. The F1s of DEEP and SIMPLE are 0.24 and 0.27, respectively, which are significantly smaller than F1s they obtained on other types of datasets, such as 0.68 and 0.52 on Small-L.

Large imbalanced datasets. Our experiments show that SIMPLE achieves better F1 scores than DEEP on larger imbalanced datasets. Specifically, SVM outperforms BERT on FUNNY and LR outperforms BERT on BOOK. Considering that imbalance tackling techniques potentially affect the comparisons, we conducted new experiments to calibrate the classification threshold for each model (a popular technique). Figure 7 shows comparison results after cali-

(a) FUNNY   **(b)** BOOK
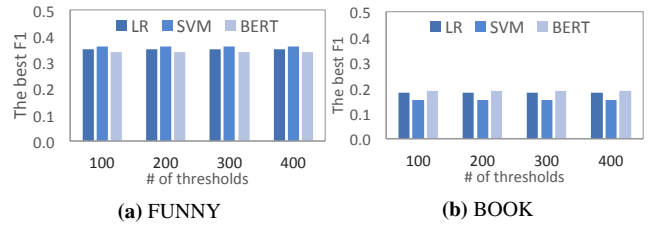
Figure 7: The best F1 of LR, SVM, and BERT on FUNNY/BOOK (1 million labels) by varying classification thresholds

bration. The results show that simple models still show better F1s on FUNNY and similar F1s on BOOK, in comparison to BERT. Notably, We observe that adding calibration improves F1s of all models, especially BERT, but does not change the tendency that simple models perform similar or even better than BERT. More experiments are presented in Appendix of the technical report [26].

6.2 Effect of dataset characteristics

In this section, we conduct analyses to understand the effects of dataset characteristics on the tagging quality. Our analyses consider the size of training set, the skewness of label ratio, and the existence of informative tokens.

6.2.1 Size of training set

To understand how the size of training set affects tagging quality, we increase the number of training data for LR, SVM, and BERT and measure their F1s. We use AMAZON, YELP, FUNNY, and BOOK as representative datasets, given that they contain abundant labels. We fix the number of test data to 100,000, since we do not observe significant differences in tagging quality when using more test data.

We plot the F1s of LR, SVM, and BERT with regard to the increasing numbers of training data in Figure 8. As expected, all models obtain F1 improvements when getting more training data, especially for LR and SVM. Increasing the training size of AMAZON from 2000 to 20,000 improves the F1 of LR (from 0.79 to 0.86), SVM (from 0.80 to 0.89), and BERT (from 0.90 to 0.93), respectively. The F1 improvements of LR and SVM (0.07 and 0.09) are greater than BERT (0.03). Similarly, the F1 improvements of LR and SVM on YELP (0.04 and 0.05) are higher than BERT (0.02), suggesting that increasing training size to promote F1 is more effective on simple models than deep models. In other words, the performance gap between simple models and deep models narrows with increase in training size. On FUNNY and BOOK, all models achieve smaller tagging F1s than that they achieve on AMAZON and YELP. To better show the differences, we set the maximal F1s to 0.3 as shown in Figure 8c and Figure 8d. No matter whether the training size is small or large, BERT does not show a superior F1 compared with LR and SVM. Along with the increase of training size, BERT performs similarly to LR and even worse than SVM on FUNNY, whereas it performs similarly to LR and slightly better than SVM on BOOK.

Notably, although LR and SVM achieve similar or even better F1s than BERT in some cases, they require numerous labels at the scale of tens of thousands. This is a big concern since data labeling may rely on human annotation rather than rule generation. Given that most of real-world datasets are on a small scale, BERT is still an appealing option.

Vocabulary size analysis. To explain that why enlarging the training set increases the tagging quality, we calculate the numbers of
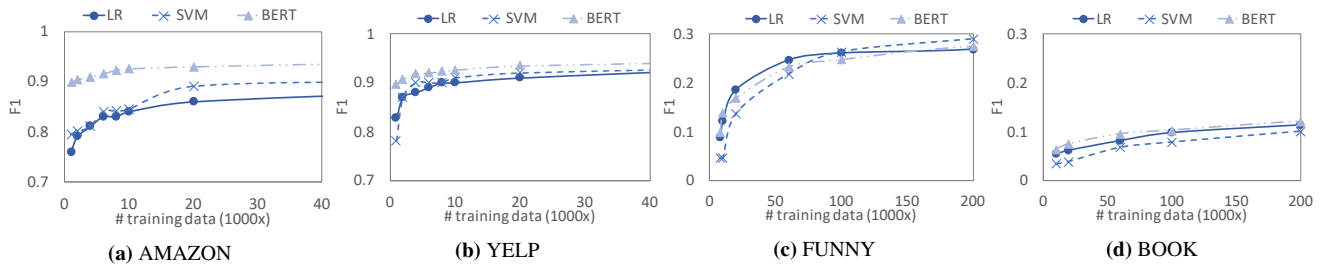
Figure 8: F1 of LR, SVM, and BERT by increasing the size of training data
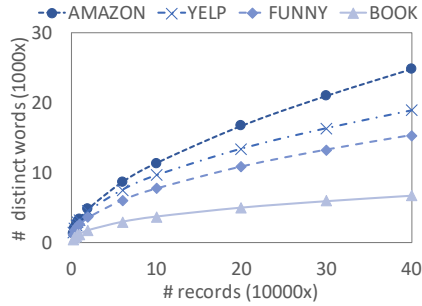

Figure 9: The number of distinct words keeps increasing when expanding the training set

distinct words (i.e. vocabulary size) as we feed more records to expand a training set. We depict the increment of distinct words in Figure 9. On each of AMAZON, YELP, FUNNY, and BOOK, the number of distinct words keeps increasing, leading to more words being included in training processes. This explains why F1 becomes higher when the training size increases. We also observed that F1s of all three models change negligibly after the training size reaches a certain number (e.g., 100,000 on YELP), indicating that the promoting effects of training size on F1 are limited when the sizes reach certain thresholds.

6.2.2 Label ratio

We evaluate how the ratio of positive instances affects the F1s of tagging models. To perform the evaluation, we adopt AMAZON, YELP, FUNNY, and BOOK as training datasets and adjust their ratios of positive instances from 10% to 90%. For each ratio r, we randomly sample 100,000 records from each dataset. Within these 100,000 records, the number of positive instances is r of 100,000 and the number of negative instances is $1 - r$ of 100,000. The records are randomly split into two parts, of which 80% of the records are devoted to training set and 20% are devoted to testing set. As a result, all models obtain higher F1s on 4 datasets when the ratios of positive instances increase, suggesting that higher ratios of positive instances bring tagging models towards higher F1s. This result is shown in Figure 10.

We find that F1 improvements regarding the increase in the ratio of positive instances exhibit two trends. First, they are more significant when the ratio of positive instances is small ($< 25\%$), while more flattened when the ratio is large ($\geqslant 25\%$). For example, F1 of BERT gains 0.06 improvement on AMAZON when the ratio increases from 10% to 20%, but only 0.01 when the ratio changes from 80% to 90%. Second, F1 improvements on AMAZON and YELP are less obvious than those on FUNNY and BOOK. For example, when the ratio increases from 10% to 20%, F1 improvements of BERT on AMAZON and YELP are 0.05 and 0.04, while F1 improvements on FUNNY and BOOK are 0.12 and 0.16. This

result suggests that increasing label ratio to promote F1 is more effective on dirty datasets (FUNNY and BOOK) than clean datasets (AMAZON and YELP).

Besides F1 improvements, we also observe that the F1 gaps between LR/SVM and BERT decrease when the ratio of positive instances increases. On AMAZON, F1 gap between LR and BERT decreases from 0.16 to 0.03 when the ratio increases from 0.1 to 0.9. Similarly, F1 gap on YELP decreases from 0.06 to 0.01 when the ratio goes up. The corresponding decreases in F1 gap may due to more unseen positive instances that are brought by the dataset with an increasing ratio. These unseen positive words have more influences on LR and SVM than on BERT as LR and SVM rely more on the occurrence of words for the tagging task.
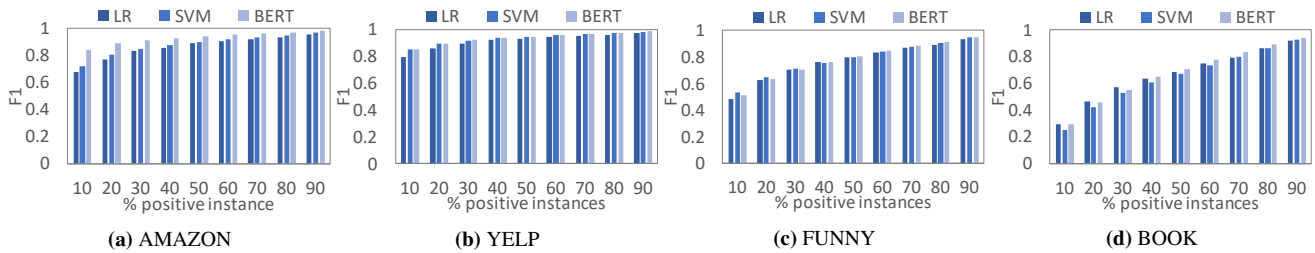
6.2.3 Informative tokens

The F1s of a tagging model on different datasets can be significantly various, even when these datasets have the same training size and label ratio. For example, both AMAZON and BOOK* have more than 1 million labeled instances and present a percentage of positive instances of 50%. However, BERT achieves 0.96 F1 on AMAZON and 0.74 F1 on BOOK* (shown in Figure 1), between which the F1 gap is 0.22. We also observe similar behaviors of other models on different datasets, such as the F1 gap of SVM on YELP and FUNNY* is 0.15.

We assume that tagging labels on FUNNY* and BOOK* are harder because these two datasets are dirtier in labels. To understand what ingredients in a dataset make it hard to be tagged, we analyze informative tokens that can separate positive records from negative records. First, we try to identify such tokens from the datasets. To perform the identification, we calculate the percentage of positive instances containing token t (denoted as P) and the percentage of negative instances containing t (denoted as N). P and N measure the occurrence of t in records concerning two different types of labels. We then sort the $P - N$ values of all tokens in AMAZON, YELP, FUNNY*, and BOOK* in descending order. Thereafter, we present the top 5 informative tokens and their frequencies of occurrence in table 8. The result shows that top tokens from AMAZON and YELP are all sentiment words that express positive opinions, such as "great" and "love". These words semantically link to the task itself, i.e. tagging positive sentences. However, top tokens from FUNNY* and BOOK* contain some stop words such as "that" and "on". These stop words appear in high frequencies in both positive and negative records.

6.3 Towards a higher F1

Our study can serve as a reference for practitioners to understand how F1 is affected. To make sure that our study can be utilized, we summarize and visualize our results as a heat map (Figure 11). We select BERT and SVM as the representative deep model and simple model, respectively. We describe their training size, ratio of positive instances, label quality of individual dataset, and tagging F1s

(a) AMAZON **(b) YELP** **(c) FUNNY** **(d) BOOK**

Figure 10: F1 of LR, SVM, and BERT on the 4 large datasets sub-sampled (100k) with different label ratios.

Table 8: Informative tokens regarding each dataset and their frequencies of occurrence in **P**ositive and **N**egative instances. We sorted all tokens by **P** − **N** descendingly and presented the top 5 tokens. Top tokens on AMAZON and YELP are all sentiment words. Top tokens on FUNNY* and BOOK* include stop words such as "that".

AMAZON	P	N	YELP	P	N	FUNNY*	P	N	BOOK*	P	N
great	0.27	0.09	great	0.39	0.15	that	0.75	0.41	he	0.13	0.06
love	0.15	0.05	delicious	0.14	0.02	you	0.69	0.37	that	0.25	0.19
best	0.12	0.04	love	0.17	0.06	on	0.70	0.40	she	0.13	0.08
easy	0.08	0.02	friendly	0.16	0.06	of	0.87	0.58	her	0.13	0.08
well	0.16	0.11	best	0.19	0.09	at	0.60	0.31	him	0.06	0.03

Dataset	Size	Ratio	Quality	BERT	SVM	F1
SUGG	9K	0.26	clean	0.86	0.77	0.96
HOTEL	8K	0.05	clean	0.67	0.55	0.85
SENT	11k	0.10	clean	0.57	0.51	0.75
PARA	7K	0.17	clean	0.65	0.59	0.64
HOMO	2K	0.71	clean	0.95	0.89	0.53
HETER	2K	0.71	clean	0.93	0.87	0.42
TV	13K	0.53	clean	0.81	0.68	0.32
EVAL	10K	0.38	clean	0.81	0.73	0.21
REQ	10K	0.18	clean	0.84	0.69	0.10
FACT	10K	0.36	clean	0.82	0.69	
REF	10K	0.02	clean	0.93	0.79	
QUOTE	10K	0.02	clean	0.66	0.10	
ARGUE	23K	0.44	clean	0.78	0.72	
SUPPORT	23K	0.19	clean	0.54	0.45	
AGAINST	23K	0.24	clean	0.62	0.51	
FUNNY	5M	0.03	dirty	0.32	0.38	
BOOK	18M	0.03	dirty	0.15	0.15	
AMAZON	4M	0.50	clean	0.96	0.93	
YELP	560K	0.50	clean	0.96	0.96	
FUNNY*	244K	0.50	dirty	0.82	0.81	
BOOK*	1M	0.50	dirty	0.74	0.7	

Figure 11: F1s of BERT and SVM on 21 different datasets

in Figure 11. To visualize F1 values, we present small F1 (< 0.53) as blue color and deepen color when values decrease, and present large F1 ($\geqslant 0.53$) as red color and deepen color when values increase. By retrieving our heat map, practitioners can estimate the approximate tagging F1s for their datasets and choose the appropriate tagging model to obtain better tagging F1s.

Our study indicates that practitioners should try BERT if their targeting datasets are small. They can expect F1 improvement as much as 0.56. However, F1 improvement of BERT will be less significant if a dataset has a large number of labels. When tagging the datasets with both large sizes and high ratios of positive instances, such as FUNNY, BOOK, AMAZON, YELP, FUNNY*, and BOOK*, BERT does not show appealing F1 improvements while taking plenty of training time (several days). In this case,

practitioners may consider simple models like SVM as an alternative choice, given that SVM can achieve a similar F1 value while taking much less training time, in comparison to BERT.

In addition to choosing appropriate tagging models, practitioners should also pay attention to dataset selection. Our study shows that AMAZON and YELP enable high F1s regardless of tagging models. These two datasets own abundant training data, exhibit balanced ratios, and include clean labels. HOMO and HETER also allow tagging models to achieve high F1s. Although these two datasets are smaller in size compared with AMAZON and YELP, they have more positive instances ($> 70\%$) than negative instances. The higher label ratio makes the semantic tagging easier. In contrast, some datasets like FUNNY and BOOK can hinder tagging models from obtaining high F1s. The reason is that their labels are dirty and ratios of positive instances are small. Therefore, practitioners should be cautious when preparing datasets and try to get datasets with large size, high ratio of positive instances, and pure cleanliness (e.g. no missing annotations).

7. CONCLUSION

Our study is the most comprehensive one that used the largest number of real-world datasets to compare deep models and simple models. Our results reveal for the first time that dataset characteristics are the key factors to determine whether deep models can achieve better tagging quality than simple models. Given the raw complexity of real-world datasets, choosing a suitable tagging model for a specific dataset rather than sticking with deep models should be the way of performing tagging tasks in the future. Our study, especially the visualized heat map will be the most informative instruction for practitioners to choose a suitable tagging model for their dataset, by considering its scale, label ratio, and cleanliness.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] C. C. Aggarwal and C. Zhai. A survey of text classification algorithms. In *Mining Text Data*, pages 163–222. 2012.

[2] M. J. A. Berry and G. Linoff. *Data mining techniques - for marketing, sales, and customer support*. Wiley computer publishing. Wiley, 1997.

[3] V. Blinov, V. Bolotova-Baranova, and P. Braslavski. Large dataset and language model fun-tuning for humor recognition. In *ACL*, pages 4027–4032, 2019.

[4] J. L. Boyd-Graber, K. Glasgow, and J. S. Zajac. Spoiler alert: Machine learning approaches to detect social media posts with revelatory information. In *ASIST*, pages 1–9, 2013.

[5] A. Cattle and X. Ma. Recognizing humour using word associations and humour anchor extraction. In *COLING*, pages 1849–1858, 2018.

[6] Y. D. Challenge. https://www.yelp.com/dataset/documentation/main.

[7] Y. Chang, C. Hsieh, K. Chang, M. Ringgaard, and C. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *JMLR*, 11:1471–1490, 2010.

[8] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*.

[9] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[11] Y. Diao, H. Lin, L. Yang, X. Fan, D. Wu, D. Zhang, and K. Xu. Heterographic pun recognition via pronunciation and spelling understanding gated attention network. In *WWW*, pages 363–371.

[12] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer, 2001.

[13] GLUE Benchmark. https://gluebenchmark.com/leaderboard. 2019.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[15] I. Guy, A. Mejer, A. Nus, and F. Raiber. Extracting and ranking travel tips from user-generated reviews. In *WWW*, pages 987–996, 2017.

[16] F. E. Harrell Jr. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] X. Hua, M. Nikolov, N. Badugu, and L. Wang. Argument mining for understanding peer reviews. In *NAACL-HLT*, pages 2131–2137, 2019.

[19] J. H. Huggins, T. Campbell, and T. Broderick. Coresets for scalable bayesian logistic regression. In *NIPS*, pages 4080–4088, 2016.

[20] O. Ivanciuc et al. Applications of support vector machines in chemistry. *Reviews in computational chemistry*, 23:291, 2007.

[21] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98*, pages 137–142, 1998.

[22] S. Kim and E. H. Hovy. Automatic identification of pro and con reasons in online reviews. In *ACL*, 2006.

[23] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[24] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[26] J. Li, Y. Li, X. Wang, and W.-C. Tan. Deep or simple models for semantic tagging? it depends on your data. https://arxiv.org/abs/2007.05651. *arXiv preprint 2007.05651*, 2020.

[27] B. Liu. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[28] J. Liu, S. Wang, and Y. Sun. Olenet at semeval-2019 task 9: BERT based multi-perspective models for suggestion mining. In *SemEval@NAACL-HLT*, pages 1231–1236, 2019.

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[30] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[31] T. Miller, C. Hempelmann, and I. Gurevych. Semeval-2017 task 7: Detection and interpretation of english puns. In *SemEval@ACL*, pages 58–68, 2017.

[32] C. Mitcheltree, V. Wharton, and A. Saluja. Using aspect extraction approaches to generate review summaries and user profiles. In *NAACL-HLT*, pages 68–75, 2018.

[33] A. Morales and C. Zhai. Identifying humor in reviews using background text sources. In *EMNLP*, pages 492–501, 2017.

[34] S. Negi, K. Asooja, S. Mehrotra, and P. Buitelaar. A study of suggestions in opinionated texts and their automatic detection. In **SEM@ACL*, 2016.

[35] S. Negi and P. Buitelaar. Towards the extraction of customer-to-customer suggestions from reviews. In *EMNLP*, pages 2159–2167, 2015.

[36] S. Negi, T. Daudert, and P. Buitelaar. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *SemEval@NAACL-HLT*, pages 877–887, 2019.

[37] S. Novgorodov, G. Elad, I. Guy, and K. Radinsky. Generating product descriptions from user reviews. In *WWW*, pages 1354–1364, 2019.

[38] L. Poddar, W. Hsu, and M. Lee. Author-aware aspect topic sentiment model to retrieve supporting opinions from reviews. In *EMNLP*, pages 472–481, 2017.

[39] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, et al. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30, 2016.

[40] P. Potash, A. Romanov, and A. Rumshisky. Semeval-2017 task 6: #hashtagwars: Learning a sense of humor. In *SemEval@ACL*, pages 49–57, 2017.

[41] S. S. Pradhan, W. H. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *HLT-NAACL*, pages 233–240, 2004.

[42] Questia.
https://www.questia.com/library/controversial-topics. 2020.

[43] V. Rakesh, W. Ding, A. Ahuja, N. Rao, Y. Sun, and C. K. Reddy. A sparse topic model for extracting aspect-specific summaries from online reviews. In *WWW*, pages 1573–1582, 2018.

[44] SEMEVAL2020. International workshop on semantic evaluation. 2020.

[45] Sklearn Linear SVM. https://scikit-learn.org/stable/modules/generated/sklearn.svm.linearsvc.html. 2020.

[46] C. Stab, T. Miller, B. Schiller, P. Rai, and I. Gurevych. Cross-topic argument mining from heterogeneous sources. In *EMNLP*, pages 3664–3674, 2018.

[47] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[48] H. F. Transformers. https://huggingface.co/transformers/.

[49] M. Wan, R. Misra, N. Nakashole, and J. J. McAuley. Fine-grained spoiler detection from large-scale review corpora. In *ACL*, pages 2605–2610, 2019.

[50] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[51] S. Wang, N. Phan, Y. Wang, and Y. Zhao. Extracting API tips from developer question and answer websites. In *MSR*, pages 321–332, 2019.

[52] I. Weber, A. Ukkonen, and A. Gionis. Answers, not links: extracting tips from yahoo!answers to address how-to web queries. In *WSDM*, pages 613–622, 2012.

[53] D. Yang, A. Lavie, C. Dyer, and E. H. Hovy. Humor recognition and humor anchor extraction. In *EMNLP*, pages 2367–2376, 2015.

[54] Z. Yu, J. Tan, and X. Wan. A neural approach to pun generation. In *ACL*, pages 1650–1660, 2018.

[55] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, pages 15–28, 2012.

[56] L. Zhang and B. Liu. *Sentiment Analysis and Opinion Mining*. 2017.

[57] X. Zhang, Z. Qiao, A. Ahuja, W. Fan, E. A. Fox, and C. K. Reddy. Discovering product defects and solutions from online user generated contents. In *WWW*, pages 3441–3447, 2019.

[58] X. Zhang, J. J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657, 2015.

[59] D. Zhu, T. Lappas, and J. Zhang. Unsupervised tip-mining from customer reviews. *Decision Support Systems*, 107:116–124, 2018.

[60] Y. Zou and W. Lu. Joint detection and location of english puns. In *NAACL-HLT*, pages 2117–2123, 2019.