Strength in Numbers: Using Big Data to Simplify Sentiment Classification

Apostolos Filippas1 and Theodoros Lappas2,*

Abstract
Sentiment classification, the task of assigning a positive or negative label to a text segment, is a key component of mainstream applications such as reputation monitoring, sentiment summarization, and item recommendation. Even though the performance of sentiment classification methods has steadily improved over time, their ever-increasing complexity renders them comprehensible by only a shrinking minority of expert practitioners. For all others, such highly complex methods are black-box predictors that are hard to tune and even harder to justify to decision makers. Motivated by these shortcomings, we introduce BigCounter: a new algorithm for sentiment classification that substitutes algorithmic complexity with Big Data. Our algorithm combines standard data structures with statistical testing to deliver accurate and interpretable predictions. It is also parameter free and suitable for use virtually "out of the box," which makes it appealing for organizations wanting to leverage their troves of unstructured data without incurring the significant expense of creating in-house teams of data scientists. Finally, BigCounter’s efficient and parallelizable design makes it applicable to very large data sets. We apply our method on such data sets toward a study on the limits of Big Data for sentiment classification. Our study finds that, after a certain point, predictive performance tends to converge and additional data have little benefit. Our algorithmic design and findings provide the foundations for future research on the data-over-computation paradigm for classification problems.

Keywords: sentiment analysis; big data; classification; opinion mining

Introduction
The ability to automatically label the sentiment of a given text segment as positive, negative, or neutral is a fundamental component of mainstream applications such as reputation monitoring,1 sentiment summarization,2 review mining,3 recommender systems design,4 and modeling consumer behavior.5 Relevant literature typically refers to this task as sentiment classification or sentiment analysis.6,7

The popularity of sentiment classification has motivated a significant body of work and has led to the design of numerous algorithms.8,9 A study of the relevant literature in chronological order reveals that these algorithms are becoming significantly more complex with time. Early work primarily focused on simple lexicon-based approaches, which were then extended by incorporating basic linguistic features.10,11 The next stage brought about the use of increasingly complex machine learning algorithms that formulate sentiment classification as a supervised learning task.6,12 The use of Natural Language Processing (NLP) techniques for feature engineering further improved the results of this approach, while also increasing its complexity.13,14 Today, the state-of-the-art tries to unlock the power of deep learning: a branch of machine learning based on modeling abstract relationships in unstructured text via multilayer graphical models, such as artificial neural networks15 and advanced language constructs, such as word embeddings.16

Even though new techniques are consistently pushing the performance boundaries in sentiment classification, their ever-increasing complexity has multiple drawbacks that we address in our work:

1IOMS Department, NYU Stern School of Business, New York, New York.
2School of Business, Stevens Institute of Technology, Hoboken, New Jersey.

*Address correspondence to: Theodoros Lappas, School of Business, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, E-mail: tlappas@stevens.edu

Downloaded by New York Univ/7214371 from online.liebertpub.com at 09/22/17. For personal use only.
The view was definitely a bonus. Did not live up to standards. Jet planes also landing nearby.

The rooms were clean and sunny. Parking is not very convenient. We asked for a different room.

This place is a hidden treasure. The food is not worth the wait. I called reception twice, no response.

Very reasonable prices. It’s full of cockroaches. They gave us extra time to check out.

Easily accessible and very clean. They go above and beyond. Parking was extra.

The soup was yummy. Million buck view. They leave little chocolates on your pillow.

We loved the food. The place has an eclectic feel. This was our fourth stay here.

The most boring game ever. Hats off to the chef. I was told the food would take 5 minutes.

Best experience of my life. The location cannot be beat. This was our fourth stay here.

Interpretability: Complex machine learning methods typically have very limited interpretability; even though we can prove that they perform well by testing them on new data, their nonlinear nature makes their outcomes hard to explain to decision makers.17

Parameter tuning: Complex algorithms are notoriously hard to tune, as the vast number of parameters prohibits manual tuning and requires exhaustive or “intelligent”—but still computationally expensive—methods for automatic tuning.18,19

Specificity: Existing methods are designed for classifying entire documents. As a result, their accuracy suffers when they have to classify smaller segments, such sentences or phrases that bear very little evidence in terms of vocabulary or context.

Infrastructure cost: Algorithmic complexity translates to steep rises in computational costs, as well as in software and hardware costs.20–22

Recruitment cost: Firms willing to accept the high infrastructure costs must inevitably also invest in top quality talent that is able to build, manage, and utilize this infrastructure. In practice, however, limited funds and steep competition in the talent market have turned the acquisition of technical talent into a challenging endeavor, especially for small- and medium-sized firms.23–26

The negative consequences of the increasingly complex algorithms for sentiment classification motivate us to consider a different approach. Table 1 presents a set of sentences that are easy, medium, and hard to classify as positive or negative. Even a simple, lexicon-based approach that counts the number of known positive or negative words, these examples do not include obviously positive or negative expressions. Instead, the polarity of these sentences is highly context dependent. For instance, the sentence This was our fourth stay here, which we extracted from a hotel review, informs us that the reviewer has repeatedly visited the hotel in the past. Given that a customer is highly unlikely to return four times to a business that she is not pleased with, we intuitively expect this sentence to carry positive sentiment. Another example is the phrase Minutes to downtown, which reveals the hotel’s close proximity to a specific location. The fact that a city’s downtown area

### Table 1. Sentences of easy, medium, and high difficulty for sentiment classification

<table>
<thead>
<tr>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>The food was great.</td>
<td>We will stay longer next time.</td>
<td>Minutes to downtown.</td>
</tr>
<tr>
<td>Best experience of my life.</td>
<td>The location cannot be beat.</td>
<td>They have shuttles if needed.</td>
</tr>
<tr>
<td>The most boring game ever.</td>
<td>Hats off to the chef.</td>
<td>I was told the food would take 5 minutes.</td>
</tr>
<tr>
<td>We loved the food.</td>
<td>The place has an eclectic feel.</td>
<td>This was our fourth stay here.</td>
</tr>
<tr>
<td>The soup was yummy.</td>
<td>Million buck view.</td>
<td>They leave little chocolates on your pillow.</td>
</tr>
<tr>
<td>Easily accessible and very clean.</td>
<td>They go above and beyond.</td>
<td>Parking was extra.</td>
</tr>
<tr>
<td>Very reasonable prices.</td>
<td>It’s full of cockroaches.</td>
<td>They gave us extra time to check out.</td>
</tr>
<tr>
<td>This place is a hidden treasure.</td>
<td>The food is not worth the wait.</td>
<td>We asked for a different room.</td>
</tr>
<tr>
<td>The rooms were clean and sunny.</td>
<td>Parking is not very convenient.</td>
<td>Jet planes also landing nearby.</td>
</tr>
<tr>
<td>The view was definitely a bonus.</td>
<td>Did not live up to standards.</td>
<td></td>
</tr>
</tbody>
</table>
is typically a popular destination (especially for hotel guests) implies a positive sentiment. At a glance, the sentence *Jet planes also landing nearby* does not carry any sentiment. However, its negative polarity becomes apparent if we consider that it comes from a hotel review. Specifically, the noise that inevitably comes with the hotel’s proximity to an airport is clearly undesirable for the hotel’s guests. One could argue that proximity to the airport enhances the hotel’s accessibility and could motivate positive comments. However, such a comment would be unlikely to include the terms *jet planes* and *landing*. Instead, we would expect a positive proximity comment to be similar to the one that we discussed previously (e.g., *Minutes to the airport*).

The dependency of the polarity of these examples on context-specific information makes them particularly challenging to address. Standard lexicon-based or supervised methods are unlikely to be effective, even if they are extended via the use of advanced feature engineering. Furthermore, the state-of-the-art in deep learning has only recently started to explore textual representations that take context into account.\(^{27,28}\) Even though the initial findings of such efforts are very promising in terms of performance,\(^ {29,30}\) they also introduce yet another layer of complexity, and demand familiarity with advanced concepts such as artificial neural networks and word embeddings.\(^ {16,31}\) These obstacles limit the (informed) use of such methods to the minority of practitioners and firms that possess the necessary experience, infrastructure, and skills. In addition, the complexity of such methods essentially eliminates the interpretability of their results, especially for nonexperts.

The examples in Table 1 verify that more challenging sentences require more complex algorithms. As we get closer to simulating the way in which humans process and generate natural language constructs, it is reasonable to expect that our algorithmic machinery will get larger and more complex. What if we could take a radically different, purely agnostic approach that is seemingly oblivious to the context or thought process behind a statement? Let us again consider the phrase *Minutes to downtown*. Suppose that we have a random sample of five English hotel reviews that include the phrase *Minutes to downtown*. Out of these five reviews, four are positive and one is negative. This small sample provides some evidence that the phrase is positive. However, some of the occurrences of this phrase may be random and have little to do with the review’s overall rating. Our confidence is thus limited due to the small size of the sample. If, for instance, our sample consisted of 9700 positive and 300 negative reviews, then our confidence would be much higher. Similarly, if our sample consisted of 5050 positive and 4950 negative reviews, a safer prediction would be that the statement is neutral. What if we had a sample of 100,000 or even a sample of 1 million reviews? Previous work has explored the value of Big Data for predictive tasks.\(^ {32–34}\) Intuitively, we expect convergence to a confident prediction to occur after a certain sample size.

In this work, we apply this simple count-based paradigm to design the BigCounter algorithm for sentiment classification. Given a very large corpus that includes both positive and negative documents of arbitrary length (e.g., customer reviews), BigCounter predicts the sentiment of a given short text sequence \(s\), such as a sentence or phrase, by using a simple statistical test to compare the positive and negative counts of \(s\) in the corpus. To account for the fact that some sequences might not occur frequently enough to allow for a confident test, we extend BigCounter to count the frequency of flexible wildcard patterns extracted from \(s\). The algorithm delivers an accuracy that competes and often surpasses the state-of-the-art.

The simple design of the BigCounter algorithm provides it with the following advantages:

1. It adopts a simple algorithmic approach based on standard data structures and statistical testing, leading to minimal software requirements and a very lightweight implementation.
2. It is parameter free and suitable for use virtually “out of the box,” which makes it appealing for organizations wanting to leverage their troves of unstructured data without incurring the significant expense of creating in-house teams of data scientists.
3. It produces easily interpretable predictions that can be traced back to actual examples from the input data set.
4. It can accurately classify sentences, even if it is trained on data labeled just at the document level. Our comparisons with two benchmark algorithms demonstrate its advantage on real data sets.
5. It is naturally parallelizable and thus scalable to very large data sets.

Our methodology has implications for practitioners in both academia and industry, as it offers a simple and effective alternative to the increasingly complex methods for sentiment classification. In addition, our methodology lowers the barrier to entry for organizations that
want to mine their growing text repositories but cannot afford the infrastructure and talent required by state-of-the-art machine learning algorithms. Furthermore, our study on the limits of Big Data can help managers make informed decisions about how much data their firm needs to achieve accurate classification results. Finally, our work lays the foundations for future research on the use of similar methods for multilabel classification tasks in other domains.

**Background and Related Work**

We begin this section with an overview of extant methods for sentiment classification. We then discuss the motivation and theoretical background of our own approach.

**Sentiment classification**

Sentiment analysis (also known as opinion mining) is a broad field that covers multiple tasks relevant to extracting opinions from different types of unstructured text, such as customer reviews, articles, and blog posts. Arguably the most prevalent of these tasks is that of classifying a given text segment as positive or negative, which is also the focus of our own work.

The underlying theme of all previous relevant methods is their effort to simulate the way human authors generate text to encode their sentiment. Lexicon-based approaches were the earliest successful attempts in this direction. These methods predict the polarity of a text by counting the number of known positive and negative words that it includes. At its core, this approach utilizes two lexicons of positive and negative words, which the practitioner needs to provide as input. Several extensions were subsequently proposed, such as domain-specific and automatically constructed lexicons. NLP techniques can be used to capture linguistic constructs that cannot be addressed by simple lexicons or extract semantic information that can improve an algorithm’s prediction. For instance, previous work has combined lexicons with linguistic constructs such as negation rules (e.g., “not good”), rules that enhance or change a word’s sentiment (e.g., “very good”), intra- and intersentence conjunctions, synonyms, and antonyms. Lexicon-based approaches are intuitive, easy to implement, and can also be competitive if properly customized for the domain of application. However, they require considerable tuning and fail to predict statements that do not include leads from the underlying lexicons, such as those that we showed in the second and third columns of Table 1.

A second family of methods formulates sentiment classification as a standard supervised learning problem. In this setting, the input to the method consists of a set of training instances with known labels. Given the training data, a machine learning algorithm builds a predictive model that is then used to classify new instances. One of the main benefits of this approach is that it allows us to experiment with a wide range of established classification algorithms, such as Naive Bayes, Logistic Regression, and Support Vector Machines (SVMs). In the absence of sufficient manually annotated data, the interested practitioner can use a semisupervised algorithm to extend the training set with automatically labeled instances.

Even though supervised learning techniques can be very competitive in the context of polarity prediction, their simplifying assumptions prevent them from accurately predicting challenging instances that use context and complex linguistic constructs to express sentiment. Arguably the most influential assumption is that the order of the words in a document does not affect its overall sentiment, which allows algorithms to treat the document as a bag of words. This assumption greatly simplifies the process of building predictive models, but also sacrifices the valuable information that comes with the order of the words, such as idioms with a clear positive or negative sentiment. This information can be partially salvaged via the use of NLP techniques to improve feature engineering. Specifically, rather than using single words as tokens, one could define complex features such as n-grams (e.g., New York, buffalo chicken wings), or combinations of neighboring words (e.g., a binary feature that encodes whether or not the words airport and noise appear in the same sentence).

Even though such features can indeed lead to performance improvements, feature engineering is a complex task that involves manual tuning and the consideration of an arbitrarily large number of candidate features.

The state-of-the-art from the domain of machine learning includes methods from the exciting area of deep learning. These methods use advanced graphical models, such as different variants of neural networks, to model various levels of abstractions over the input data. Artificial neural networks are inspired by the neuron structure in the human brain, and their ultimate goal is to model the brain’s decision-making and processing functions. One of the benefits of deep learning methods is the utilization of advanced word representations that go far beyond single words or even simple linguistic features. Arguably the most characteristic example is the use of word embeddings: continuous word representations based on the assumption...
that words in similar contexts have similar meanings.\textsuperscript{31,44} Deep learning research has evaluated neural networks in the context of various domains, including sentiment classification. For instance, convolutional neural networks have been used for polarity prediction.\textsuperscript{36} Such models introduce a single layer of convolution over a set of word representations obtained via previously proposed unsupervised neural language models.\textsuperscript{15,31} For the same task, good performance is obtained by a variant of the standard recursive neural network (RNN), referred to as a recursive neural tensor network,\textsuperscript{29} that allows for direct interactions between the continuous representations (embeddings) of the words in the considered vocabulary. This facilitates the detection of meaningful linguistic constructs, such as negation. Neural network have also been customized for sentiment classification by incorporating sentiment scores into word embeddings.\textsuperscript{45} Additional graphical models for polarity prediction include gated recurrent neural networks\textsuperscript{46} and adaptive recursive neural networks.\textsuperscript{47}

Previous work has repeatedly verified the superiority of deep learning methods for sentiment classification. However, their performance comes at the cost of significant complexity and decreased interpretability.\textsuperscript{48} Neural networks are typically treated as “black boxes,” as the large number of interacting nonlinear parts make it difficult to understand exactly how they function and to interpret their results.\textsuperscript{17} This can create resistance to adoption of these techniques in business settings, especially in highly regulated industries with decision makers who lack the expertise required to fully comprehend the inner workings of such complex models.\textsuperscript{49} This type of criticism precedes the emergence of deep learning, as it dates back to the early days of artificial neural networks.\textsuperscript{50–52}

Our approach: memory over computation

The shortcomings and ever-increasing complexity of existing methods motivate us to present a novel and much simpler algorithm for sentiment classification. Consider a random English sentence $S$ that we need to classify as positive, negative, or neutral. Our training corpus $D$ consists of documents (e.g., customer reviews) that have been manually annotated as positive or negative. Rather than try to reverse engineer $S$, we opt to treat $S$ as a single object and simply count the number of times that it appears in a positive and in a negative document from $C$. If the difference in the two counts is statistically significant, we mark $S$ with the majority label. Otherwise, we mark it as “neutral.” Admittedly, if the frequency $N_{S,C}$ of $S$ in $C$ is low, then our confidence in the prediction will also be low. Our confidence rises as $N_{S,C}$ becomes larger, and we expect convergence to occur after a certain point. For instance, $N_{S,C} = 1,000,000$ likely does not lead to more accurate predictions than $N_{S,C} = 100,000$ or even $N_{S,C} = 10,000$.

Our “memory over computation” approach is motivated by the usage-based paradigm for language learning, which posits that children develop their language skills by initially memorizing and gradually refining simple patterns.\textsuperscript{53–55} For example, after memorizing the pattern “Where is the X?” the child can then customize it by substituting $X$ for other terms or constructs that represent meaningful objects such as “book” or “cookie jar.”

In this setting, children learn how to properly select which constructs to insert into each pattern based on frequency of usage. For instance, while the child is likely to hear the phrase “Where is the cookie?” often, she is unlikely to hear the phrase “Where is the eat?” The first phrase will thus be validated via observation and repetition, while the second one will be rejected. This mechanism allows the pruning of the endless patterns and combinations that can theoretically exist in a language.

The usage-based paradigm has strong ties to extensive theoretical work on formulaic sequences and their effects on language learning.\textsuperscript{56,57} A formulaic sequence is a continuous or noncontinuous sequence of words that is stored and retrieved from memory at the time of use, and is not subject to generation or analysis by the language grammar.\textsuperscript{58} Formulaic sequences offer processing efficiency because single memorized units, even if made up of a sequence of words, are processed more quickly and easily than the same sequences of words generated creatively.\textsuperscript{59,60} In other words, it is easier to memorize prefabricated chunks of language that can be used on-demand, rather than to build a new sequence by considering vocabulary and grammar rules.\textsuperscript{61–63}

The success of our approach comes down to a simple question that we address in this work: do we have enough data to confidently predict the polarity of any possible formulaic sequence? At a glance, the response to this question is negative, as the number of possible sentences is simply overwhelming and it would require an unrealistically large training corpus. However, as we demonstrate in our study, we can drastically reduce this data dependency and achieve highly accurate predictions via the use of flexible patterns that allow the replacement of words with wildcards. For instance, if the frequency of the pattern “the * cannot be beat” in the input corpus is large enough to allow for a confident prediction of its
polarity, then we can memorize this pattern and use it to classify any matching sentences (e.g., “the location cannot be beat,” “the food cannot be beat”).

This flexible representation is similar in spirit to that of the “schema theory” introduced by Holland and served as the basis of numerous follow-up works on genetic algorithms. According to schema theory, one can cover a large part of a multidimensional space via a schema that defines constraints on the defining dimensions. For instance, a general schema would restrict one of the dimensions to a specific value while allowing the others to assume any value. While this schema would cover a very large part of the search space, it is likely too general to represent a meaningful pattern that applies to many valid points. Hence, by balancing the restrictiveness-applicability tradeoff, we can discover interesting rules that accurately and succinctly represent our data. As we discuss in detail in The BigCounter Algorithm section, our algorithm faces a similar tradeoff, as it utilizes textual patterns that include both fixed terms and wildcards and can therefore match multiple sentences. In the context of sentiment classification, the goal is then to mine patterns that can match many sentences of the same (negative or positive) polarity and few or no sentences of the opposite polarity.

The BigCounter Algorithm
In this section, we present the BigCounter algorithm for sentiment classification. Our method belongs to the broad class of supervised machine learning algorithms and operates in three steps: preprocessing, training, and prediction.

Preprocessing
The input to the preprocessing phase is a large collection of weakly annotated documents, that is, documents that are annotated as positive or negative at the document level but not necessarily at the sentence level. BigCounter begins by segmenting each document \( D \) into its sentences. For each sentence \( S \), the algorithm then computes \( P_S \): the set of all possible patterns generated if we replace every possible subset of words in \( S \) with wildcards (*) that represent any word. As we discuss in the following section, the use of wildcards allows BigCounter to predict the sentiment of sentences that have a very low or even zero frequency in the training corpus.

If \( D \) has a positive (negative) label, then we increment the positive (negative) count of every pattern \( P \in P_S \) by 1. We demonstrate this with an example in Figure 1. In this example, we focus on the sentences “I would definitely return” and “I would never return,” mined during the preprocessing phase from a positive and negative review, respectively. The figure presents all the possible flexible patterns that can be generated from these two sentences. The middle column includes the patterns that the two sentences have in common. For each of these common patterns, we increment their respective positive and negative counts by 1. For the patterns in the left and right columns, we increment only the positive and negative count, respectively.

Training
The input to the training phase consists of a collection of distinct wildcard patterns, as well as a positive and negative count for each pattern. The training phase
uses the two counts to assign a positive or negative label to the pattern.

We formally model the assignment as a two-sided coin toss, where the possible outcomes are positive or negative. Given a pattern \( P \), let \( p_P \) denote the true probability of a positive outcome and let \( N_P^+, N_P^- \) be the pattern’s positive and negative counts, respectively. We want to decide whether the two counts provide enough evidence to reject the neutrality hypothesis \( H_0 : p_P = \frac{1}{2} \) which states that \( P \) is equally likely to occur in positive and negative texts, and hence bears no sentiment. The appropriate statistical test for this task is the binomial test, which examines whether the deviations of the distribution of observations with two possible classes from the theoretically expected distribution are statistically significant. We also account for the fact that positive reviews are known to be much more common than negative reviews by updating \( p_P \) to reflect the class proportions in the data set.\(^{68}\)

If our test rejects the neutrality hypothesis, then we assign the majority label to \( P \) and record it in a simple key-value (pattern-label) store, which we refer to as the Polarity-Index. The algorithm repeats this process for all the patterns extracted from all the documents in the training corpus. We present the pseudocode for BigCounter’s preprocessing and training phases in Algorithm 1.

**Prediction**

Prediction with BigCounter is a straightforward task. Let \( S \) be a sentence whose sentiment we want to determine. The prediction process begins by extracting the set of flexible patterns \( P_S \). The algorithm then retrieves the polarities of the patterns from \( P_S \) that are contained in the index \( I \). If the index includes multiple matching patterns, BigCounter simply labels \( S \) with the majority label. If the numbers of positive and negative matching patterns are equal, or if no matching patterns exist, then there is insufficient evidence that \( S \) bears sentiment, and it is labeled as neutral.

One can consider different aggregation policies by assigning weights to the predictors that match a sentence. For example, matching flexible sequences with fewer wildcard characters may be assigned higher weights. In our experiments on the algorithm’s predictive accuracy, we use unweighted majority voting, as we found alternative policies to be less competitive.

\(^{69}\) For large values of \( N_P^+ + N_P^- \) we can use the faster \( \chi^2 \) test to closely approximate the binomial test.

**Discussion**

One of the main benefits of BigCounter is that it can deliver accurate sentence-level predictions even when trained on instances with document-level annotations. The large volume of product reviews on online marketplaces is an inexpensive source of such data. In contrast, other state-of-the-art approaches, such as recursive neural networks, often require sentence-level or even phrase-level annotations.\(^{29}\)

On the contrary, BigCounter naively assumes that all the patterns mined from an overall positive (negative) document also carry positive (negative) sentiment. While positive reviews will generally include positive patterns, there are some cases where this assumption is violated. For instance, only part of the text in a customer review actually carries sentiment while the rest covers objective information, positive sentences are used sarcastically in negative reviews, or an otherwise positive review includes a single negative comment. Such inevitable occurrences are likely to contaminate the positive and negative counts of each pattern. Despite such contaminations, given a very large data set of reviews with many occurrences of a truly positive pattern, the pattern will appear in more positive than negative reviews (the opposite is the case for negative patterns). Finally, the difference between the positive and negative counts of neutral patterns will not be statistically significant (as determined by the binomial test), and thus, the pattern will not be added to the Polarity-Index.

**Scalability Enhancements**

In this section, we describe three techniques that enable us to efficiently apply our method on very large data sets.

---

**Algorithm 1: BigCounter Training.**

**Input:** Corpus of training docs \( D \), confidence level \( \alpha \)

1: \( U = \emptyset \) \hfill \( \triangleright \) Set of unique wildcard patterns
2: \( \text{pos} = \emptyset, \text{neg} = \emptyset \) \hfill \( \triangleright \) Key-value stores for the pos and neg counts of each pattern
3: for each doc \( D \in D \) do
4: for each sentence \( S \) of \( D \) do
5: Generate the set of wildcard patterns \( P_S \) from \( S \).
6: for each pattern \( P \in P_S \) do
7: Add \( P \) to \( U \)
8: if \( D \) is positive then \( \text{pos}[P]++ = 1 \) \hfill \( \triangleright \) \( \text{pos}[P], \text{neg}[P] \) are initialized to 0
9: else \( \text{neg}[P]++ = 1 \)
10: n
11: \( \text{Polarity-Index} = \emptyset \) \hfill \( \triangleright \) An empty key-value store
12: Set \( p \) equal to the percentage of positive docs in \( D \)
13: for each pattern in \( U \) do
14: if BinomialTest(pos[\( P \)], neg[\( P \)], \( p \)) \( < \alpha \) then
15: if \( \text{pos}[P] > \text{neg}[P] \) then Polarity-Index \( \{ P \} = + \)
16: else Polarity-Index \( \{ P \} = - \)
17: Return Polarity-Index
Efficiently performing many statistical tests

The BigCounter algorithm performs a large number of statistical tests during training. By exploiting a simple structural property of the binomial test, we can significantly speed up this phase. Consider a pattern $P$ with $N_P^+ + N_P^- = n$. The binomial test will label $P$ as positive if $N_P^+$ is sufficiently large. Intuitively, there should exist some number $\bar{x}$ such that we can reject the neutrality hypothesis and label the sequence as positive if and only if $N_P^+ > \bar{x}$.

We prove that such a number exists and that we can efficiently compute it.\(^\dagger\) Let $\alpha$ denote the confidence level that we are using in our statistical tests and let $p = \frac{N_P^+}{n}$. The bound $\bar{x} = x(N_P^+, n, x)$, the solution to the following problem:

\[
\begin{align*}
\min_x & \quad x \\
\text{s.t.} & \quad \sum_{k=x}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leq \alpha \\
& \quad x \in \{0, 1, \ldots, n\}.
\end{align*}
\]

The optimization problem (P1) is nonlinear and discrete. However, observe that as the value of $x$ decreases, the left-hand side of the inequality constraint strictly increases since more positive terms are included in the summation. Therefore, the optimal solution can be obtained following a greedy procedure: search through the solution space starting at $x=n$ and decrease the value of $x$ until a nonfeasible solution $x^*$ is reached, at which point set $\bar{x} = x^* + 1$. In the case that P1 is infeasible, the proposed procedure outputs $\bar{x} = n + 1$, a bound that never rejects the neutrality hypothesis.

For a given class probability $p$ and confidence level $\alpha$, we are now able to precompute these bounds for different values of $n$ and store them in a simple key-value store, such as a dictionary or hash map. This substitutes statistical tests (which in the case of the binomial test involve computationally cumbersome factorials) with fast memory lookups, thus dramatically speeding up the training phase. Furthermore, it is straightforward to apply our analysis to obtain similar bounds for the $\chi^2$ test, which can be used instead if the number of observations $n$ is sufficiently large.

Stopword homogenization

The notion of stopwords is used in linguistics to refer to the most commonly used words in a language. For the English language, the list of stopwords includes words such as “a,” “the,” “this,” “be,” and “and,” which account for close to 40% of the words of all written text.\(^70\)

We utilize stopwords to reduce the memory requirements of the BigCounter algorithm. During preprocessing, every stopword is replaced by the special token $\hat{s}$. We refer to this step as stopword homogenization. The homogenization step has a significant computational impact, as it greatly reduces the number of generated patterns. For example, the sentences “The price cannot be beat” and “This price cannot be beat” would both be homogenized to “\$ price cannot \$ beat” and would thus generate the exact same set of patterns. Stopword homogenization has a significant computational impact, as it reduces the memory footprint of BigCounter by about 75%.

Parallelizability and computational cost

Building the Polarity-Index can be a computationally strenuous task, especially if we want to scale up to real-life Big Data applications. Next, we demonstrate that the index-building process is highly parallelizable and can be completed via parallel threads on any distributed infrastructure.

Consider a pattern $S$ of length $n$. The first observation is that only sequences of the same length as $S$ may contribute in computing the positive and negative counters $N_P^+, N_P^-$ for the pattern. Therefore, we can build the index for each distinct length independently and in parallel. The second observation is that two sentences of the same length can generate the same pattern only if they include stopwords in the exact same positions. For example, the sentence $S = \text{“Great price for the quality”}$ has stopwords in the third (for) and fourth (the) positions. Therefore, the set of patterns $P_S$ that it generates will have the stopword token $\hat{s}$ in these positions. It follows that a sentence with a different allocation of stopwords cannot generate any of the patterns in $P_S$. We conclude that sequences with different stopword allocations are independent with respect to the statistical tests that BigCounter conducts during its training phase. Based on these two observations, our implementation of BigCounter assigns a separate thread for each batch of sentences that share the same length and the same stopword structure, efficiently parallelizing the training phase.

It is important to note that even a single-thread implementation can deal with large data volumes in a reasonable amount of time. For all three data sets that we utilized in our evaluation, a single-thread version

---

\(^\dagger\)The bound for labeling the pattern as negative can be computed in the exact same way.
of BigCounter required between 0.0005 and 0.0008 seconds per sentence. This translates to about 30 seconds for 50,000 sentences and 4 hours for 24 million sentences. The decisive factor is the length of the sentence, which is also a point of natural parallelizability for BigCounter. The average time required for a sentence of 5, 10, 15, and 20 words is 0.00041, 0.00047, 0.00087, and 0.0338 seconds, respectively. As we discuss in detail in the Data Sets and Setup section, most of the sentences in large real data sets have approximately 10 words.

The times that we report here are based on a single-thread Python implementation running on a Linux system with 32 GB of RAM and 1.2 GHz clock speed. The times achieved by a distributed implementation, such as the one that we described above, depend on the number of available processing units. Python is an attractive choice, as it provides libraries that can directly support the components of BigCounter, such as statistical testing. It also provides competitive options for parallel processing on either multiple CPUs or multicore CPUs. That being said, the simplicity of the BigCounter algorithm makes it a good candidate for other mainstream programming languages with similar capabilities, such as C, C++, and Java.

Evaluation

In this section, we present the experiments that we conducted toward the evaluation of the BigCounter algorithm. We begin with a brief overview of our data sets. We then evaluate the predictive accuracy of BigCounter by conducting extensive tests that include comparisons with the state-of-the-art. We conclude our evaluation with a study on the limits of Big Data for sentiment classification. All the data sets and software implementations used in this section are openly accessible or can be made immediately available on request.

Data sets and setup

Raw data. We utilize three data sets of reviews from the TripAdvisor, Yelp, and Amazon websites, which we crawled over the period between September 2015 and December 2015. TripAdvisor reviews pertain to the hotel industry, Yelp reviews to the restaurant industry, and Amazon reviews are focused on books. The challenge in the book domain stems from its highly subjective nature, as well as the richness of the language used in book reviews. The main benefit in utilizing three different data sets is that we can assess the generalizability of our results, across both websites and review domains. Our data sets include the text and a rating from 1 to 5 stars for each review. We assume that reviews with 4 or more stars are positive and reviews with 2 or less stars are negative.

Table 2 provides an overview of our data sets. Positive reviews are by far the most frequent category in all three data sets, confirming the existence of a strong positive bias.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Positive</th>
<th>Negative</th>
<th>Neutral</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotels</td>
<td>4,048,966</td>
<td>619,954</td>
<td>727,662</td>
<td>5,396,582</td>
</tr>
<tr>
<td>Restaurants</td>
<td>2,873,114</td>
<td>236,121</td>
<td>537,497</td>
<td>3,646,732</td>
</tr>
<tr>
<td>Books</td>
<td>1,166,161</td>
<td>185,000</td>
<td>134,705</td>
<td>1,485,866</td>
</tr>
</tbody>
</table>

Table 2. Basic data set statistics: Rating distribution

We then repeat the process for all three domains, but this time restrict the sampling process to sentences that do not include known positive or negative words (e.g., great, amazing, horrible). The identity of such words is verified via their presence in the established lexicons introduced in an article. The elimination of

---


2Due to the apparent triviality of the annotation task for human annotators, only a handful of sentences were actually discarded.

sentences with obvious sentiment cues leads to a much more challenging test set, in which the authors use more complex linguistic patterns to express sentiment. This second sampling phase leads to three additional data sets (1 per domain) with 500 annotated sentences per data set. We refer to this second corpus as SentenceCorpus-Hard. In total, our evaluation includes \((3 \times 500) + (3 \times 500) = 3000\) annotated sentences.

**Sentiment classification**

In this experiment, we assess the ability of BigCounter to predict the sentiment of sentences, and compare it with other state-of-the-art techniques. We compare our algorithm against two competitive baselines: SVMs and RNNs. SVM-type algorithms are based on an intuitive idea: given the representation of the training set examples in the feature space, the goal is to find the hyperplane that maximizes a distance measure between the different classes. SVMs have met success across a wide range of real-life applications, including sentiment classification.\(^7\) We use the multiclass SVM implementation of Scikit-learn, which applies sequential minimal optimization-type decomposition method for the training phase.\(^7\) We tune the algorithm’s predictive performance by combining cross-validation with an extensive grid search over the hyperparameter space, as well as other linguistic and syntactical feature extraction options, such as varying n-gram lengths.\(^\dagger\dagger\)

The second baseline belongs to the class of deep learning algorithms. More specifically, we use the Stanford CoreNLP sentiment analysis implementation.\(^2\)\(^9\)\(^,\)\(^7\)\(^5\) The algorithm is a sentence-level model, which begins by utilizing a pipeline of NLP techniques that include lexical and syntactical sentence parsing, tagging, and construct identification, to construct a tree representation of the sentence structure. In this implementation, the parse trees are constructed from a publicly available movie review data set,\(^2\)\(^6\) all unique phrases are extracted from the trees and require manual annotation. A RNN is then trained on top of the extracted linguistic structure and annotations. We used a grid search to optimize the parameters of the implementation, as we did for the SVM baseline.

All three algorithms are evaluated on the SentenceCorpus and SentenceCorpus-Hard corpora, which include sentences with three different labels (positive, negative, and neutral). We present the results of our experiments on the SentenceCorpus and SentenceCorpus-Hard corpora in Figures 3 and 4, respectively. The \(y\)-axis represents the achieved accuracy, while each bar on the \(x\)-axis represents a different sentence length (i.e., number of words). We report the results for each length separately to study the consistency of the three algorithms.

For the SentenceCorpus corpus, BigCounter consistently outperforms both baselines for most sentence lengths in the hotels and restaurants domains. For all three domains, BigCounter is very competitive and often the winner for sentences that include up to 8–10 words. The RNN demonstrates a slight advantage for longer sentences, although this advantage is not consistent and the top approach tends to vary across data sets and sentence lengths. We anticipated a decrease in BigCounter’s accuracy for longer sentences, although this advantage is not consistent and the top approach tends to vary across data sets and sentence lengths. This is also supported by Figure 2, which verifies that the availability of sentences steadily decreases after their length surpasses 8–10 words. We examine and quantify the effect of data availability in the following section, where we discuss our study on the limits of Big Data for sentiment classification.

\(^\dagger\dagger\)See also http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (last accessed August 16, 2017).
Even though, as anticipated, the accuracy of all three algorithms is lower for the SentenceCorpus-Hard, our algorithm performs at a high level and often surpasses the two baselines by a wide margin. This verifies that BigCounter does not require obvious sentiment cues to deliver accurate predictions, as it utilizes its own Polarity-Index: an extensive and diverse library of both obvious and nonobvious sentiment-bearing patterns. On the contrary, both the SVM and RNN baselines perform significantly worse for this data set, across domains and sentence lengths. Table 3 reports the accuracies of the three classifiers.

Testing the limits of big data for sentiment classification

In this section, we attempt to contribute to the body of research that examines performance gains from bigger data by focusing on how the size of the training corpus affects the predictive performance of BigCounter. Our approach builds on previous work that has utilized data from different domains to conduct a learning curve analysis and gauge the benefits of larger training corpora for predictive tasks.33,34 In our own evaluation, we examine the effect of increasing the size of the training set on the following:

1. The training set’s linguistic diversity, as encoded by the number of unique wildcard patterns, as well as by the number of patterns that are accepted into BigCounter’s Polarity-Index.
2. The predictive performance of BigCounter.

We generally expect that adding more data should increase the number of unique and indexed patterns. However, we hypothesize that as the training set gets larger, it becomes harder to locate never-before-seen patterns, and even harder to locate new sentiment-bearing patterns.

![FIG. 3. Predictive accuracy as a function of sentence length, evaluated on the SentenceCorpus corpus.](image)

![FIG. 4. Predictive accuracy as a function of sentence length, evaluated on the SentenceCorpus-Hard corpus.](image)
bearing patterns that make it into the Polarity-Index. This would lead to a convergence in terms of accuracy, as the algorithm would not have additional ways to classify new sentences. The purpose of this experiment is to verify the existence of such a convergence point.

We begin by randomly sampling 50,000 sentences to serve as the initial training set. We then iteratively double the size of the training set by adding a new random sample of sentences. After each addition, we recompute the number of unique patterns, the size of the Polarity-Index, and the accuracy of BigCounter on both the SentenceCorpus and SentenceCorpus-Hard corpora.

Linguistic diversity. We report the results of the first two metrics in Figure 5. We observe that, for all three domains, additional data have a major effect on both the number of generated patterns and the size of the index early in the process. This effect dwindles for larger training sets, as large amounts of additional data are required for only small increases in the number of generated patterns. This verifies that the majority of distinct language patterns are already present in smaller data sets. In addition, the index size exhibits the same diminishing returns behavior, with most of the sentiment-bearing patterns having been detected early in the process. This provides us with strong evidence that, with respect to discovering new sentiment-bearing patterns, there exists a threshold above which additional data quickly become less valuable.

A second observation is that, for all three domains, there exists a considerable difference (three orders of magnitude) between the size of the Polarity-Index and the number of unique patterns mined from the training set. The same holds for the rate at which these two quantities grow. We conclude that even though there are a great number of patterns in the reviews, the number of sentiment-bearing patterns is a lot smaller and converges a lot faster. This finding implies that the practice of acquiring additional data does not deliver substantial gains after a certain point in terms of linguistic diversity.

Accuracy. We report the results of the experiment in Figure 6. We first observe that, as expected, the effect of additional data is most pronounced during the first steps of the learning curve analysis: relatively small increases in the training set size result in large performance gains. Increases in predictive performance persist throughout the learning curve analysis, but soon start to diminish to the point that they become marginal. This implies that, after some point, additional data have little value, at least with respect to sentiment classification. It is worth noting the overlap of the two curves for the books domain; this is expected due to the fact that reviews from this domain contain less obvious leads and hence are more challenging to classify.

BigCounter requires a substantial amount of data to construct the index. While the algorithm is competitive with smaller amounts of data, we observe that there are increasing (but diminishing) returns to additional data across all metrics we examined. As expected, the required data set size is proportional to the linguistic diversity of the application: the detection of rarer patterns.

### Table 3. Summary of the classification accuracies

<table>
<thead>
<tr>
<th>Domain</th>
<th>Type</th>
<th>BigCounter</th>
<th>RNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotel</td>
<td>Easy</td>
<td>0.743</td>
<td>0.654</td>
<td>0.504</td>
</tr>
<tr>
<td>Hotel</td>
<td>Hard</td>
<td>0.620</td>
<td>0.443</td>
<td>0.409</td>
</tr>
<tr>
<td>Restaurants</td>
<td>Easy</td>
<td>0.759</td>
<td>0.656</td>
<td>0.658</td>
</tr>
<tr>
<td>Restaurants</td>
<td>Hard</td>
<td>0.577</td>
<td>0.444</td>
<td>0.436</td>
</tr>
<tr>
<td>Books</td>
<td>Easy</td>
<td>0.683</td>
<td>0.736</td>
<td>0.587</td>
</tr>
<tr>
<td>Books</td>
<td>Hard</td>
<td>0.671</td>
<td>0.467</td>
<td>0.409</td>
</tr>
</tbody>
</table>

RNN, recursive neural network; SVM, Support Vector Machine.
expressions demands more data. A direct implication is that the size of the Polarity-Index varies between applications, although the number of unique patterns generated seems to converge to the same number across domains (Fig. 5).

Domain transferability
A desirable property for learning algorithms is domain transferability: given that a model is trained on data from a domain A, how does the model perform when applied to a test set of some other domain B? In practice, transferability becomes more important in the absence of sufficient training data for a specific domain of interest B. Accurate classifications are still possible in such cases, as long as the practitioner has access to sufficient data from another domain A that can complement or even substitute the training corpus. We assess the predictive accuracy of BigCounter when it is trained and tested on different domains. We report the results for the SentenceCorpus and SentenceCorpus-Hard data sets for different sentence lengths in Figures 7 and 8, respectively.

Not surprisingly, the best accuracy is consistently obtained when the training and testing domains are the same. The semantic similarity of the restaurant and hotel domains is captured by the fact that the predictive accuracy of the respective indices is very close. In fact, while the in-domain overall predictions are higher, there are even a few cases where restaurants and indices outperform each other in the out-of-domain prediction. On the contrary, the results verify that the books domain is highly dissimilar to the other two. We observe that the book-trained index achieves poor performance when used for out-of-domain prediction. Furthermore, both the hotel and restaurant indices perform substantially worse when used on the books testing sets.

Our experiments provide some preliminary evidence on the transference learning properties of BigCounter. We find that, if sufficient training data from a particular domain A are hard to obtain, practitioners can complement their data troves with data from semantically similar domains, for which data are more abundant. This observation connects our work with the extensive literature on semantic similarity and suggests interesting directions for future work.

Implications and Directions for Future Work
Our methodology for sentiment classification departs from the standard approach of trying to mathematically explain natural language. Instead, we demonstrate how the ever-increasing complexity of state-of-the-art methods can be replaced by mining weakly annotated Big Data. Our experimental evaluation against competitive baselines verifies the efficacy of this new and much simpler approach. In addition, we utilize our methodology toward a detailed study on the limits of Big Data for sentiment classification. Our study is motivated by the hypothesis that, after a certain point, adding more data to the training set does not increase performance. Our findings provide strong evidence in support of this hypothesis and deliver valuable insight on the connection between data size and performance.

Implications
Our work has implications for practitioners in both academia and industry, as it presents an intuitive alternative
to the increasingly complex and computationally expensive algorithms for sentiment classification. In addition to being much simpler while achieving highly competitive results, our approach delivers interpretable predictions that can be easily communicated to managers and decision makers, even if they do not possess an extensive technical background. This is a significant advantage over state-of-the-art algorithms, such as recent advancements in deep learning, that are typically treated as black boxes and mystify nonexperts.

Furthermore, our methodology lowers the barrier to entry for firms that want to incorporate sentiment classification into their product or data analysis tasks. This is especially important for firms that cannot afford the hardware/software infrastructure and talent required to train and tune complex computational models. Lowering a firm’s dependency on technical talent can be a significant advantage, especially for smaller firms that do not have the resources to be competitive in the ongoing talent wars. Such firms can greatly benefit by our data-over-computation paradigm and utilize data that they already have (or can easily acquire) rather than try to design and implement an algorithmic engine that surpasses their capabilities.

Finally, our study on the limits of Big Data can help managers make informed decisions about how much data their firm needs to achieve accurate sentiment classification results. The ability to make such decisions is valuable, as additional data typically come with additional acquisition and management costs, measured in both monetary terms and work hours. Therefore, a firm can achieve significant savings by not trying to crunch more data than it actually needs to. Our study can help managers and team leads strategically design their data acquisition efforts by revealing the type of data that they need to acquire (e.g., in terms of the origin domain, vocabulary, polarity) to complement their training set, cover previously uncovered cases, and achieve more accurate results.

Directions for future work
Future research can consider applying our approach to different domains. Even though the focus of our work...
is on sentiment classification, our methodology can also be applied to any document classification task, as well as to the task of labeling specific sentences within a larger document. For instance, consider the problem of assigning topic labels to tweets. Rather than depending on elusively training data sets with tweet-level annotations, a practitioner could utilize our approach on weakly annotated data, such as batches of tweets from users with known topical interests (e.g., we expect politicians to tweet about politics and athletes to tweet about sports).

Another promising direction would be to combine our methodology with the state-of-the-art in text representation. This would include the use of a neural network-based method, such as the one suggested by Mikolov et al., to map each sentence to a low-dimensional space. We can then build an index with the polarity of each sentence embedding, using the same approach that we currently apply for wildcard patterns. We can then predict the sentiment of a new sentence by (1) computing its embedding, (2) finding its nearest indexed neighbors within the low-dimensionality space, and (3) assigning the majority label among the neighbors. These embedding-based labels could then replace or complement their pattern-based counterparts, depending on their effectiveness in different contexts.

It is our hope that our findings and methodological contributions will inspire and support relevant research in this domain and will motivate the design of simple but effective algorithms that can mine actionable insights from Big Data.

**Author Disclosure Statement**

No competing financial interests exist.

**References**