Privacy-Preserving Patent Search: Additively Homomorphic Encryption Techniques for Private Text Mining over Public Datasets

Jennifer Katherine Fernick[1,2,3] and Sebastian Verschoor[1,2]

[1] Institute for Quantum Computing, Waterloo, Canada
[2] David R. Cheriton School of Computer Science, Waterloo, Canada
[3] Centre for Applied Cryptographic Research, Waterloo, Canada
jkfernic@uwaterloo.ca, srverschoor@uwaterloo.ca

Abstract. Machine learning on encrypted data is an emerging field that can be used for a variety of real-world applications. A particular application might be where a user has an idea for an innovative technology, and would like to confidentially submit their idea to an external platform that can semantically analyze their idea and return whether this idea has already been patented, without anyone other than the user being privy to her idea. The generalization of such a construction would be when a user wishes to make a private query to a public or third-party dataset, while maintaining the confidentiality of their query. In this paper, we make use of somewhat-homomorphic privacy- preserving building blocks for machine learning to enable this type of analysis, while extending and optimizing known theoretical constructions.

Keywords: Machine Learning, Encrypted Data, Text Mining, Data Mining, Natural Language Processing, Semantic Analysis, Patent Search, Cryptography, Additively Homomorphic Encryption, Privacy-Preserving Data Mining.

1 Introduction

In this paper, we construct a system for text- mining which allows a user to submit as input an encrypted patent idea to a patent database, and determine whether her idea has already been patented, receiving references to related patents as output. In contrast to existing techniques for patent search, this one preserves the confidentiality of the user's search such that neither the server nor a third- party adversary will be able to learn the user's unpatented idea.

The major contributions of this paper are: (1) To offer techniques for secure text mining based on known primitives for machine learning on encrypted data; (2) To implement these techniques and offer an application enabling users to confidentially validate the uniqueness of their intellectual property against a patent database; (3) To replicate the results of an important and elegant paper on machine learning over encrypted data [1]; and (4) Extending and optimizing upon the techniques suggested in prior works [1].

This paper presents prior art and our present efforts across several sections. In Section 2, we outline our specific application scenario, highlight the motivation for this work, and present our security model. In Section 3, we offer prerequisite knowledge on machine learning, and in Section 4, a literature review on the convergences between machine learning/data mining, natural language processing, and computing on encrypted data. In Section 5, we articulate our research objectives and demonstrate our set-up, implementation, and results. In Section 6, we discuss our findings, and in Section 7, we offer directions for future work.

2 Introduction

2.1 Scenario

We can imagine a scenario where a user has an idea for an innovative technology, and they would like to know whether there are any existing patents or other intellectual property restrictions upon their idea. To learn this, they must find all similar ideas to theirs from within a database of patents. This is difficult for a few reasons. Firstly, the database of patents is large and they would like to use efficient computational methods to narrow down the amount of physical searching and consequent reading that they, the user, must perform. Secondly, the English keywords representing their idea might have semantic similarity to other keywords that are used for a relevant patent, but this patent will be overlooked if they do not query using the particular keyword used in the documentation of the patent. Thirdly, since their idea is valuable and patentable, and the database is a large and frequently-updated online repository, they would like to be able to perform their patent search without revealing the words that they are using to perform the search.

2.2 Motivation

This case study is motivated by a more general problem, which is that the status quo of information on the internet is that there are large volumes of useful data, but an absence of privacy and security in interacting with these datasets and the data mining tools that can efficiently operate upon them.

This problem can be generalized to the following: A user (client) would like to perform a text-based search of a database (server), that both keeps secret the contents of their search, as well as improves relevance of matches by finding close-proximity keywords and finds database entries containing those similar keywords.

2.3 Security Model

Stating a precise security model is essential to making meaningful statements about mining on encrypted data. In our model, the user (client) trusts themselves and their local computing environment, and can safely assume that the encryption scheme is semantically secure against a polynomial-time adversary, and that no parties beside the user themself has access to the user's private keys. They can assume that the server is

semi-honest (honest- but-curious), and that any attacks mounted on the system by the server or by a third party are poly- time computationally bounded.

3 Preliminaries

3.1 Machine Learning

For computing to be powerful in complex, real-world systems, it must be able to successfully extract relevant and concise insights from complex datasets. Since machine learning offers powerful techniques for performing this analysis on distributions of data, it is increasing in social and technical importance. Machine learning algorithms take as input a collection of samples ("training" data) and build models that capture some [hopefully] important relationships between features of the underlying dataset. Using this model, they can analyze future samples ("user" data) to perform actions like regression, classification, or clustering, to consequently output categorical or numerical labels or information regarding a given sample. As noted by [1], while machine learning models are necessarily more compact than the original training dataset, they also must necessarily capture some information about that dataset. As such, these models are themselves a variably rich source of information.

Machine learning can be discovered and represented using different kinds of models and techniques, depending on the type of task to be performed. To simplify for the purposes of this paper, we can consider two primary types of machine learning to exist:

1. *Supervised Learning*: This is the approach in which a labelled training set has been used to create a machine learning model, and user queries involve inputting unlabeled pieces of data and receiving the generated label for that data point as output.
2. *Unsupervised Learning*: In this approach, the dataset is unlabeled and statistical techniques are used to find patterns within the dataset. The most notable category of unsupervised learning is *Clustering*, whereby a set of unlabeled samples is grouped into several clusters, based on the similarity relations of the feature vectors of the samples.

Together, these two categories cover the primary algorithmic tools we will consider for our computational problem. But how might these tools be applied to encrypted data?

4 Literature Review

This area of research unites fields such as machine learning/data mining, natural language processing, and computing on encrypted data. In particular, prior work on text mining, machine learning on encrypted data, and statistics are relevant to our work. In this section, we begin by reviewing work that has been performed to enable machine learning algorithms to be applied to encrypted datasets. We then proceed to review

current work in text mining, and draw conclusions about how text mining may be applied in a privacy-preserving scenario.

4.1 Machine Learning on Encrypted Data

Machine learning is an increasingly ubiquitous technology which allows real-world systems to efficiently deal with complex information. Machine learning as a field represents an important step between user-initiated task-specific computing, and automated decision-making. Given the volume of data on the internet and the sizes of datasets required for modern technologies such as smart homes, self- driving cars, automated financial advisors and other forms of informational and decision-making automation, machine learning is the reason that these tasks can be performed in real-time on dynamic and complex datasets and thus that these types of technologies can even exist at all. In the context of patent searches, machine learning presents an opportunity to classify large numbers of patents and recommend similar patents, to help users better understand the intellectual property landscape, rather than being limited to string-matching on their specific input queries, which can leave many related documents unnoticed.

With all of the benefit that machine learning provides in discovering information and correlations, there often exist corresponding deficiencies in privacy-preservation or security for these systems. Relatedly, for systems where privacy and/or security come first, machine learning has been previously unavailable. Various research efforts into the performance of machine learning algorithms on encrypted data have opened new avenues for data mining, while defining and limiting compromises to security and privacy of data.

Secure Multi-Party Computation (MPC) refers to a cryptographic subfield with the objective of allowing multiple parties to compute a function over their collective inputs while maintaining the privacy of the individual inputs, such that the group members may know the result of the computation but not learn the values of the other individuals' inputs. As it is known that any distributed computing task can be securely computed [26], we can conclude that an arbitrary function – including a machine learning construction – could be built from canonical Secure MPC tools such as [21], [22], [23], [24], [25]. However, it is widely accepted in the cryptographic community that these rigorous tools are computationally expensive and likely infeasible (or at the very least, suboptimal in terms of performance) for common machine learning classifiers.

Homomorphic Encryption is a branch of secure MPC whereby operations can be performed on a ciphertext to have corresponding operations occur to the plaintext, allowing the party performing computation to compute on data for which it does not know the ciphertext [26]. (Fully) Homomorphic encryption schemes have been used to perform machine learning on encrypted data. In these constructions, a fully homomorphic encryption scheme capable of computing an arbitrary function on encrypted data is used to construct the underlying functions for specific machine learning algorithms. While these approaches have very strong security guarantees, and are quite generalizable (i.e.:

in the fully-homomorphic case, they can be used to construct an arbitrary function), they tend to be inefficient compared to all other techniques for machine learning on encrypted data. As highlighted by [1], they also suffer from unique limitations that narrow the range of learning algorithms that can be implemented fully homomorphically. These limitations include: (1) Using these schemes, one cannot encrypt arbitrary real values; (2) Post- encryption, the ciphertext size grows several orders of magnitude; (3) Fully-homomorphic techniques are computationally expensive, especially for anything multiplicative; and (4) Fully-homomorphic schemes place limits on the depth of operations to be performed, and have the consequent requirement to bootstrap.

Privacy-Preserving Approaches to Machine Learning on Encrypted Data also exist. These differ from fully-homomorphic approaches in that the functions used to compute the specific subroutines of the machine learning algorithms are generally not constructed fully-homomorphically. Instead, these constructions make use of other cryptographic tools such as somewhat homomorphic encryption or secure multi-party computation. While the somewhat-homomorphic techniques are more limited in applicability than fully-homomorphic techniques, they are generally far more efficient [1]. It is important to note that different instantiations of privacy-preserving machine learning may even have different definitions of "privacy" – some of these scenarios attempt to maintain the privacy of the training data, while a smaller number of them seek to maintain the privacy of the classification process. (In the use case presently under study, it is indeed the privacy of the classification that is our concern). In addition, security in these constructions can be a wildcard and the comparisons of a scheme's relative security to another scenario is rarely apples-to- apples. Since these techniques rely upon constructions with varying security properties, different definitions of "privacy" and are deployed in a variety of architectures to protect different subsets of information assets, there is no generic statement that can be made about their security. Rather, each construction in a given model must be evaluated in turn.

Perhaps the most interesting of these approaches are those which make use of *additively homomorphic* encryption schemes to construct reasonably-efficient building blocks for the subsequent construction of encrypted-data versions of canonical machine learning algorithms, where the statistical tools underlying these algorithms can be performed additively homomorphically. This is the technique exposited in [1], and which serves as the closest technical influence upon our present work.

A subset of privacy-preserving machine learning approaches are those that make use of differential privacy. Differential privacy is the idea that when you have two datasets (D1 and D2), differing in at most one element, the mechanism must satisfy a formula that ensures that the log of the probability of the mechanism for a given dataset divided by the log of the probability of the mechanism for the other data set falls within some privacy budget, epsilon. This essentially means that the removal of a specific individual from a dataset should not change computation on the dataset, and therefore that the individual's privacy is maintained even when they are present within the larger dataset. These constructions are special because they seek to bound information loss on a single user (or that user's data) through the addition of noise to the training/test dataset. This allows for machine learning to be efficiently performed over the dataset with small and

bounded impact upon an individual user's privacy for data that is included in the learning model. These techniques in general focus on keeping the individuals in the training/test dataset safe, and do not account for securing the classifier itself [19]. A downside to differential privacy is that there is generally a trade-off between privacy and accuracy of the model, since both are contingent upon the addition of noise. However, some constructions exist [20] that deal meaningfully with the trade-off between privacy and accuracy in differentially private learning models and offer optimization and guidance.

Metadata-based approaches. For the sake of completeness, it is worth considering one other approach to machine learning on encrypted data – namely, the approach in which the machine learning can be performed by a third party with no special cryptographic relationship with the user or their data, and the user's cryptography can be presumed secure against any polynomial-time adversary. Such approaches perform machine learning algorithms simply on the metadata of the encrypted data, which could include such features as data volume, to/from addresses, geolocation, round- trip times, or protocols used. These approaches are useful in cases where a third party wishes to learn about patterns within a specific dataset, but does not – for access, regulatory, or legal reasons – have the ability to construct the system using one of the other tools outlined in this section. This approach has been successfully deployed in real-world scenarios such as the detection of anomalous network traffic [4].

While all of these approaches have something interesting to offer the field, for our selected use case – and indeed, for most user-centered Platform- as-a-Service type of data mining on the internet – the homomorphic/privacy-preserving constructions show the most immediate applicability. These constructions would include techniques based on secure multi-party computation, fully homomorphic encryption (capable of computing arbitrary functions, but with the drawback of high computational costs for anything making use of multiplicative homomorphism) as well as somewhat-homomorphic schemes that can offer the necessary security guarantees to operate on encrypted text. We have innovated beyond existing constructions in a practical sense by applying these ideas thoughtfully to a complex systems design to solve a real-world problem, and in a theoretical sense by optimizing on the constructions in the literature.

4.2 Text Mining and Natural Language Processing

Substantial work has been performed in the field of machine learning and knowledge discovery related to mining textual data.

Text mining has been performed in the literature through the application of popular machine learning techniques onto language. Successful implementations include Naïve Bayes [5][6][7], Decision Trees [8][9], k-Nearest Neighbours [10][11][12], Support Vector Machines [13][14][15][16], and Artificial Neural Networks [17][18]. In these approaches, words in a body of text are taken as feature vectors, and are computed upon to output knowledge about a given text, such as a sentiment analysis or a classification of the text into a defined category.

Notably, all of the algorithms found in our review of general (unencrypted) text mining primitives made use of *supervised learning*, where labelled data is required for training and the output of the algorithm is (at some point) a class for the specific data instance. This presents a challenge for solving the particular machine learning problem that we had in mind – namely, that of textual similarity. As a consequence, as will be articulated later in this paper, our approach needed to involve both algorithm and broader systems design to cope with using supervised learning techniques for our use case.

5 Experiment

5.1 Research Objectives

The objective of this work is to demonstrate that secure text mining can be performed using constructions from the field of machine learning on encrypted data. This enables a more sophisticated clustering to be performed than traditional text- matching, which can improve the relevance and comprehensiveness of results.

5.2 Systems Architecture

Our system is constructed in a client-server model, where the client is the entity wishing to make a confidential search of the text database, and the server is the cloud service which contains the patent records and performs the operations over these records to return the results of the query.

5.3 Algorithms

The prior art of [1] on the development of privacy-preserving machine learning schemes through the construction of machine learning primitives using additively homomorphic encryption served as a springboard for the contributions we offer in this paper. Their solution to the problem of privacy-preserving data mining was important because they created modular building-blocks that can be used to construct a variety of canonical machine learning tools, except that the constructions they enabled would work over encrypted data. While this could in principle be done for many algorithms via fully homomorphic encryption, the authors selected to use additively homomorphic encryption, allowing for a great improvement in performance. In the context of machine learning – particularly the mining of large datasets – these performance gains are meaningful, since efficiency of computational time and space are particularly important in cases where the encryption scheme increases computational overhead, but the problem to be solved deals with high data volumes and complex models.

In our work, we started from the existing implementation of privacy-preserving Naïve Bayes [1]. However, we quickly found some problems with this construction, namely the Naive Bayes classifier requires to be initialized with pre- classified data. As a proof-of-concept in the first stage of our experimental work, we chose to pick a classification that was already done in the data, namely the CPC-classification [27].

Given the set of all classes, $\boldsymbol{C}$, the Naive Bayesian model is computed for each of the classes that are present in the training data. The feature vector space, $\boldsymbol{X}$, is the full collection of all words that are present in any of the data. For every class $\boldsymbol{c_i}$, a prior probability is computed:

$$\Pr[C = c_i]$$

Following the original work of [1], we then specified two feature values per feature $\boldsymbol{x_j}$ per class $\boldsymbol{c_i}$:

$$p = \Pr[X_j = x_j \mid C = c_i]$$

specifies the conditional probability that a word is present in the text, given that we are looking at the class $\boldsymbol{c_i}$. The other value, $\boldsymbol{(1 - p)}$, represents the conditional probability that a word is not in the text for the class.

These probabilities are estimated as follows in the training phase: The prior probability is simply the ratio of the class to the total number of entries in the training data set. Per class $\boldsymbol{i}$, the conditional probability for each word is estimated as:

$$(\texttt{word\_count\_i} + 1) / (\texttt{total\_word\_count\_j} + \texttt{unique\_words\_j})$$

One is added to each probability to avoid multiplications by zero during the classification phase and the `unique_words_j` term is added to renormalize so all probabilities add up to one.

In the regular non-encrypted setting one would classify by multiplying these probabilities. Instead, the (natural) logarithm of each probability is computed. The reason is that we can now add the values instead of multiplying them. We are allowed to do this, because we are only interested in the relative probabilities, not their absolute values. Computing over the logarithms is sometimes also done in a non-encrypted setting in order to improve numerical stability. Each log-probability is dynamically scaled and truncated so that the entire range fits in a 64-bit integer. This final step allows us to work in the plaintext space of the Paillier cryptosystem.

During the classification phase, the feature vector that the user uses to query the model is a binary vector with every entry being one if the word is in the text. Optionally, this vector can be computed in a more sophisticated manner by offline preprocessing of the query text. The dimension of the vector corresponds to the total words that are in the model.

5.4 Implementation and Optimizations

Our training and test data are sourced from the Patent Grant Full Text database on Kaggle [2]. This is a large sample of the Patent Grant Data (*Grant Red Book*) published by the United States Patent and Trademark Office (USPTO). This dataset contains a range of entry types, by for our purposes the CPC classifications and the abstract for each patent is all that is relevant. It is important to note that this dataset represents a single round of granted patent applications from which to build our proof-of- concept

– for our tool to be production quality, we would need to actively pull updates from the Grant Red Book to the server to ensure that queries led to operations over the most recent and complete version of the USPTO database.

Our machine learning primitives make use of the building blocks for Naïve Bayes developed by the authors in [1]: Computing the `argmax` function over encrypted data (using the Quadratic Residue cryptosystem of Goldwasser-Micali [28]) and computing additions over encrypted data (using the Paillier cryptosystem [29]). In [1], the authors were able to construct useful machine learning primitives for basic tasks that we were able to then use to perform textual analysis on the database using encrypted queries.

Extending their original research, however, we made novel optimizations to their construction for Naïve Bayes. Consider: classification is done by computing the formula:

$$\texttt{argmax\_}\{i \in [k]\} \{log \Pr[C = c_i] + Sum\_j^d \, log \Pr(X_j = x_j | C = c_i)\}$$

There are two simple optimizations we apply to this formula. The first is server-side, where we recognize that in the conditional probability:

$$(\Pr[X_j = x_j | C = c_i])$$

each feature can only take two values: 0 and 1. Since the probabilities add to one, we can only store the value for $X_j = 1$ and reduce the total required storage for the model to approximately half the original size. The additional advantage is that about half the amount of data needs to be sent to the client.

The second optimization comes at the client side: computing the above formula requires for every class a sum over all conditional probabilities in the model (actually by a product when translated to the Paillier cryptosystem). However, a feature vector will usually be very sparse. In that case we can benefit from not adding (multiplying) all values and instead only add the corresponding probabilities where our feature vector takes on the value one. Since the weight of the user feature vector is constant and we are only interested in the relative outcomes per class, we can apply this optimization. This speeds up the computation client-side without leaking information (the resulting sum is never decrypted and only used during the subsequent comparisons in the computation of the `argmax` function).

The source code for our work is available at [30].

5.5 Experimental Results

Below, we offer some performance results from our experiment. For qualitative results, we direct the reader to Section 6, *Discussion and Conclusions*.

Table 1: Client-side computational time savings due to data sparsity

Sparsity	Query time
100.0%*	600 ms
50.0%	480 ms

25.0%	390 ms
10.0%	350 ms
1.0%	320 ms
0.1%	290 ms
0.01%	300 ms

* where 100% sparsity refers to no optimization

Table 1 shows the computational time saved on the client side, by taking advantage of the sparsity of the data. The sparsity is expressed as the percentage of features that have value one (percentage of all words that are in the text). The first entry should not be interpreted as a query with all words, but as the non-optimized version. Here we see that even a completely random entry (50%) already has an advantage over the non-optimized version. Below a sparsity of 1%, the time to compute the Pallier product becomes insignificant compared to the time it takes to compute the `argmax` function.

Table 2: Number of patents per patent class

Class	Patents per class
A	659
B	579
C	390
D	20
E	138
F	350
G	1518
H	1713

Table 2 shows the distribution of the data. In total, the dataset contains 5367 patents, which are classified into 8 different CPC classes. The word distribution is extracted from the abstract per patent. These abstracts contain 15396 unique words, so that the feature vector has a corresponding dimension of 15396. A real-world application might want to perform some pruning to this word-list, in order to enhance performance and improve accuracy. For our proof of concept, this was not required.

Table 3: Scaling of performance with number of classes

Classes	Total time	Data transferred
8	320 ms	31.0 MB
4	260 ms	15.0 MB
2	70 ms	7.9 MB

Table 3 shows how the performance scales with respect to the number of classes. Since we have to compute a conditional probability for every class for every feature, the data

size is expected to scale linearly with both these parameters. By increasing the number of patents, the model size should not increase (although new words could be introduced increasing the number of features). Both optimizations have been applied and all entries were computed with a sparsity of 1%.

6 Discussion and Conclusions

We began this paper by seeking to understand known constructs for machine learning on encrypted data, and how machine learning techniques can be used to perform text mining. Together, these fields could enable us to offer solutions for the problem of users wishing to query a data lake without revealing to the database the content of their searches, and these searches being more complex than a traditional search – rather, they make use of machine learning techniques to do advanced data analytics such as semantic analysis.

In the course of this work, we viewed the problem through the lens of a real-world use case – whereby a user wishes to find out whether her patent idea or a semantically similar has already been claimed – but without revealing the idea to a server that may be interested in stealing the idea. In doing so, we built, optimized, and extended upon existing research to produce a software product to solve this problem.

In doing so, we have: (1) Translated machine learning techniques for secure text mining based on known primitives for machine learning on encrypted data; (2) Implemented these techniques and offer an application enabling users to confidentially validate the uniqueness of their intellectual property against a patent database; (3) Replicated the results of an important and elegant paper on machine learning over encrypted data, and provided performance benchmarks [1]; and (4) Extended and optimized upon the techniques suggested in prior works [1].

Together, these ideas have enabled us to demonstrate the validity, applicability, and performance of using the "secure building blocks" approach to constructing machine learning algorithms, and hope that this work will inspire future practical advances, in terms of both the range of building blocks and algorithms that can be constructed, as well as the use cases to which these constructions can be applied.

7 Proposals for Future Work

There exist many related directions for future work, each of which solve unique problems in machine learning over encrypted data.

The first of these directions is to build more building blocks, enabling a more complete set of machine learning techniques to be translated into version that make use of primitive functions that can be performed on encrypted data. This follows in the fine example set by [1], and would help to construct a vocabulary for future learning tasks to be expressed and performed securely.

The second of these directions is to work on a generalizable method for *combining* the machine learning building blocks for encrypted data (i.e.: to enable hybridization

or chaining of machine learning algorithms, as is generally performed in machine learning to enhance performance of an algorithm for a dataset). This was mentioned in passing in the literature with reference to AdaBoost [1], but requires more thorough analysis and results for this field to meaningfully grow and offer a functionally equivalent set of tools for the performance of learning tasks.

The third of these directions is to improve the semantic analysis of the patent database by building secure ways to take confidential user input and construct or make use of semantic networks such as Wordnet [3] to perform an additional stage of machine learning that will broaden the query to the database to include keywords semantically related to – but not exactly the same as – those initially present in the user's patent application. While this is primarily a machine learning problem, it is complexified at an algorithmic level by the requirements that: (1) the Wordnet server not learn the contents of the initial user query or the words related to it, and (2) the Wordnet semantic network transmitted to the patent server must remain confidential to all parties except the user (client).

The fourth of these tasks is a smaller one, unrelated to security but relevant to building a fully- functional system that replaces the more insecure way in which patent searches are presently performed. Specifically, this task involves building a mechanism for regularly (either via streaming or frequent batch-processing) updating the server's patent records to reflect the complete USPTO database.

The fifth of these tasks relates not to improving the translation of machine learning techniques to secure constructions, nor is it to do with solving engineering problems related to our specific use case – rather, it involves work to compare the techniques presented in this paper to other approaches that could be taken for secure text mining, including the more heavy-handed approaches making use of fully-homomorphic encryption schemes. These comparisons should study such differences as those in performance, ease of algorithm hybridization, and proposed and actual security models, and could serve as a valuable benchmark for future work in this area.

References

1. Bost, R., Ada Popa, R., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. IACR ePrint archive, https://eprint.iacr.org/2014/331.pdf, last accessed 2016/12/12.
2. Kaggle – Patent Grant Full Text, https://www.kaggle.com/uspto/patent-grant-full-text, last accessed 2016/12/12.
3. Princeton Wordnet, https://wordnet.princeton.edu, last accessed 2016/12/12.
4. Fernick, J. K., Crowley, M.: Big metadata: Machine learning on encrypted communications. RSA Conference 2017, San Francisco (2017).
5. Lee, L. H., Isa, D., Choo, W. O., Chue, W. Y.: High relevance keyword extraction facility for Bayesian text classification on different domains of varying characteristic," Expert Systems with Applications 39(1), 1147-1155 (2012).
6. McCallum, A., Nigam, K. A.: Comparison of event models for Naïve Bayes text classification. Workshop on Learning for Text Categorization, AAAI '98, 41-48 (1998).

7. Broder, A. Z., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V., Zhang, T.: Robust classification of rare queries using web knowledge. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 231-238 (2007).
8. Murthy, S. K.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery 2(4), 345-389 (1998).
9. De Comite, F., Gilleron, R., Tommasi, M.: Learning multi-label decision trees from texts and data. In: Machine Learning and Data Mining in Pattern Recognition. Springer, Berlin (2003).
10. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 42-49. (1999).
11. Han, E. H. S., Karypis, G., Kumar, V.: Text categorization using weight adjusted k-Nearest Neighbour classification. In: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '01), pp. 53-65. Springer, Berlin (2001).
12. Jiang, S., Pang, G., Wu, M., Kuang, L.: An improved k-Nearest Neighbour algorithm for text categorization. Expert Systems with Applications 39(1), 1503-1509 (2012).
13. Joachims, T.: Text categorization with Support Vector Machines: Learning with many relevant features. In: Machine Learning: ECML 1998. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence, vol. 1398. Springer, Berlin (1998).
14. Wang, T.-Y., Chiang, H.-M.: One-against-one fuzzy Support Vector Machine classifier: An approach to text categorization. Expert Systems with Applications 36(6), 10030-10034 (2009).
15. Kumar, M. A., Gopal, M.: A comparison study on multiple binary-class SVM methods for unilabel text categorization. Pattern Recognition Letters 31(11), 1437-1444 (2010).
16. Sun, A., Lim, E.-P., Liu, Y.: On strategies for imbalanced text classification using SVM: A comparative study. Decision Support Systems 48(1), 191-201 (2009).
17. Yu, B., Xu, Z.-B., Li, C.-H.: Latent semantic analysis for text categorization using neural networks. Knowledge Based Systems 21(8), 900-904 (2008).
18. Ruiz, M. E., Srinivasan, P.: Hierarchical text categorization using neural networks. Information Retrieval 5(1), 87-118 (2002).
19. Ji, Z., Lipton, Z. C., Elkan, C. Differential privacy and machine learning: A survey and review. https://arxiv.org/pdf/1412.7584.pdf, last accessed 2017/07/09.
20. Kim, D., Chen, Z., Gangapadhyay, A.: Optimizing privacy-accuracy tradeoff for privacy preserving distance-based classification. International Journal of Information Security and Privacy 6(2), 16-22 (2012).
21. Yao, A. C.: Protocols for secure computations. FOCS '82, pp: 160-164 (1982).
22. Goldreich, O., Micali, S., Widgerson, A.: How to play any mental game. SOC '87: pp 218-229 (1987).
23. Henecka, W., Kogl, S., Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: Tasty: Tool for automatic secure multi-party computations. CCS '10, pp. 451-462 (2010).
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay-secure two-party computation system. USENIX Security '04, pp. 287-302 (2004).
25. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: The sulq framework. In: Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 128-138 (2005).
26. Lindell, Y., Pinkas, B. Secure multi-party computation for privacy-preserving data mining. https://eprint.iacr.org/2008/197, last accessed 2017/07/09.

27. European Patent Office/United States Patent and Trademark Office Co-operative Patent Classification – CPC Scheme and CPC Definitions, http://www.cooperativepatentclassification.org/cpcSchemeAndDefinitions/table.html, last accessed 2016/12/12.
28. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: Proceedings of the 14th ACM Symposium on Theory of Computing (1982).
29. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '99), pp.223-238 (1999).
30. Privacy Preserving Patent Search github repo. https://github.com/sebastianv89/private-patent, last accessed 2016/12/12.