Utrecht University

MASTER'S THESIS

Semi-supervised learning for Technology Assisted Review

Author:
E. (Ercan) Öz
7974523

Project Supervisor:
Dr. A.J. (Ad) Feelders
Second Examiner:
Dr. ing. G.M. (Georg) Krempl
Daily Supervisor:
M.P. (Michiel) Bron MSc

A thesis presented for the Artificial Intelligence master

Graduate School of Natural Sciences (GSNS)
Utrecht University

June 11, 2023

Contents

Abstract

Finding all documents relevant to a specific information need in a potentially large collection of documents is essential for many researchers. This is essential not only for researchers who need to sift through thousands of studies to determine which studies are relevant for their meta-analysis but also for clinicians, policy-makers, journalists, and even the general public. Technology Assisted Review (TAR) incorporates machine learning algorithms and human feedback to find all relevant documents to achieve complete recall at a minimal cost. This study investigates methods to enhance the performance of TAR.

The availability of labeled data is often limited due to the high costs associated with labeling the data in terms of time and resources. A lack of labeled data can limit a model's capacity for generalization. Semi-supervised learning (SSL) techniques, which use unlabeled data to improve model performance, were examined to address this limitation. This thesis studies various SSL techniques for binary classification and evaluates their contributions to the TAR process.

We compared the performance of five semi-supervised learning classifiers within TAR against their supervised equivalents. The findings highlight that the semi-supervised Multinomial Naive Bayes classifier, with many-to-one correspondence via sub-topics, was able to improve the performance over its supervised counterpart multiple times, particularly in the two datasets with the lowest percentage of relevant documents. Significant improvements were also demonstrated for some datasets by combining AutoTAR and semi-supervised Multinomial Naive Bayes with sub-topics, compared to the supervised AutoTAR model. In contrast, label spreading and Support Vector Machines with self-training less frequently outperformed their supervised counterparts.

Although semi-supervised models did not consistently outperform their supervised counterparts, this research demonstrates the potential for improved performance using semi-supervised models. This was most notably observed with the semi-supervised Multinomial Naive Bayes model with many-to-one correspondence.

Keywords: Technology Assisted Review · Semi-supervised learning · Multinomial Naive Bayes · Label spreading · Support Vector Machine · Work Saved over Sampling · Self-training · Active learning

Chapter 1

Introduction and research question

1.1 Introduction

Consider the task of finding all documents relevant to an information need in a (potentially large) collection of documents. An example of such a task is scholars who have to screen thousands of studies by hand to determine which studies are relevant for their review or meta-analysis. To determine whether any given document is relevant is non-trivial. Therefore, screening a collection of documents by hand is error-prone and inefficient, as only a tiny part of the documents are relevant. This makes the data highly imbalanced. However, this is not only a problem for scholars. It is also a problem for clinicians, policy-makers, journalists, and even the general public and occurs in various domains, like electronic discovery, systematic review, investigation, research, and information retrieval evaluation (Schoot et al., 2021; Li and Kanoulas, 2020).

As screening a collection of documents by hand is error-prone and inefficient, Technology Assisted Review (TAR) is used. TAR aims to speed up the process of determining whether a document is relevant or not. The goal of TAR is to find all the relevant information given a specific information need. Missing a single relevant document could have a significant impact on, for example, a lawsuit or investigation. Therefore, achieving 100% recall is desired, preferably with low costs. The documents that TAR reviews can consist of various types of texts, like medical articles, legal documents, etc. Speeding up the process is done by incorporating machine learning algorithms and human feedback on the relevance of the documents. Currently, continuous active learning (CAL) algorithms show superior performance in efficiently finding the relevant documents in a set of documents (Li and Kanoulas, 2020).

Training data is needed to train the machine learning algorithms used in TAR. Due to rapid development in various technologies, it has become easier to collect large datasets. Typically there are more unlabeled data available than labeled data. This is because labeling data can be expensive in both time and resources. When labeled data are sparse, the model would not be strong in generalization as the amount of labeled data is often crucial for the system to generalize well. That is why utilizing the unlabeled data is an important step (Søgaard, 2013; Ding, Zhu, and Zhang, 2017).

Semi-supervised learning (SSL) does precisely this. SSL uses unlabeled data to create better models by exploiting the marginal distribution of unlabeled data. This way, the machine learning algorithms can utilize unlabeled data without manually labeling it. Because SSL has only been researched extensively in recent years, most classification algorithms still use supervised learning exclusively (Søgaard, 2013; Ding, Zhu, and Zhang, 2017).

This thesis aims to study different techniques for SSL for binary classification and evaluate the added value of these techniques in the TAR process.

The following section (Section 2) discusses TAR, the different measurements used in TAR, and the default settings. Section 3 discusses related works in which TAR and SSL are used for text classification. The different SSL methods will be discussed in Section 4. Then the experimental evaluation setup of this research will be described in Section 5. In Section 6, the results of the experiments will be shown. Finally, in Section 7, the findings and interpretation of the results are discussed.

1.2 Research question

The research question of this thesis is:

RQ. Can semi-supervised learning be used within Technology Assisted Review to improve the work saved over sampling score?

Based on the literature review (Sections 3 and 4), the SSL methods have been narrowed down to label spreading, semi-supervised Multinomial Naive Bayes (with sub-topics), and Support Vector Machine (SVM) with self-training. Label spreading is chosen based on the study of Liu, Timsina, and El-Gayar (2018). They showed that label spreading worked best in their experiments to classify medical articles as relevant or not (Section 3.4.4). The study of Nigam, McCallum, and Mitchell (2006) found high-probability models, which are then correlated with high-accuracy classifiers using Expectation Maximization. This semi-supervised Multinomial Naive Bayes with many-to-one correspondence method is chosen as their study focuses on text classification and shows much promise (Section 4.3.4). Finally, SVM with self-training is chosen as SVM is the state-of-the-art active learning approach at the moment (Ding, Zhu, and Zhang, 2017; Yu and Menzies, 2019). This way, one graph-based SSL method, one generative model, and one SVM method are compared to investigate the research question.

Chapter 2

Technology Assisted Review

Technology Assisted Review (TAR) aims to find all relevant documents, including a few non-relevant documents. This maximizes recall, while the reviewer's effort to find all relevant documents is minimized (Cormack and Grossman, 2015). The type of documents used in the TAR process can be, for example, correspondence, memos, emails, electronic business records, medical articles, legal documents, and even balance sheets (Li and Kanoulas, 2020).

2.1 Continuous active learning (CAL)

The best results of TAR are achieved through continuous active learning (CAL). First, the prediction of document relevance is made using classification or ranking algorithms. Through CAL, the documents that are most likely relevant are presented to a reviewer in batches. In turn, the reviewer assigns a label to these documents as either relevant or non-relevant. Through this process, the reviewer labels the unlabeled documents, which are then fed back to the learning method to (re-)train the ranking algorithm (Cormack and Grossman, 2015; Liu, Timsina, and El-Gayar, 2018; Li and Kanoulas, 2020).

The CAL process adds more labeled documents to the training data each iteration, improving the ranking algorithm. TAR tries to identify relevant documents based on feedback from the reviewer until all or a substantial number of relevant documents have been found. This makes it a total recall problem. In order to achieve a high recall, a depth-first search is used instead of a breadth-first search (Cormack and Grossman, 2015; Liu, Timsina, and El-Gayar, 2018; Li and Kanoulas, 2020).

The state-of-the-art active learning approaches use support vector machines (SVM). The SVM is updated after each time a human has labeled a data point to be relevant or non-relevant. The SVM then returns a new (batch of) the most relevant papers or papers that can improve the classifier most to the reviewer for labeling (Yu and Menzies, 2019).

A downside of the CAL approach is that there is much emphasis on the documents classified as most likely relevant. This greedy method could introduce a bias into the system to prefer documents similar to those found at the start of the

TAR process. This could prevent finding classes of relevant documents dissimilar to those classified as relevant at the beginning (Cormack and Grossman, 2015).

2.2 AutoTAR

AutoTAR is currently considered the state-of-the-art method for total recall tasks like TAR. AutoTAR improves recall by repeatedly selecting documents for users to review. The documents are ranked, and the most relevant documents are selected for review first (Li and Kanoulas, 2020).

2.3 One-phase workflows and two-phase workflows

There are two TAR workflows, the one-phase workflow and the two-phase workflow. With one-phase TAR workflows, the review is a single iterative process. This means that all reviewed documents are also used for training the model. Continuous active learning uses the one-phase workflow, as the model uses the relevance feedback of the reviewer on the top-ranked documents for training. In two-phase workflows, assigning different reviewers to the different phases is possible. In the legal domain, attorneys can choose to assign reviewing the documents in phase one to senior attorneys, as the labels of these documents will affect many other documents. In turn, the reviewing of documents in phase two could be done by contracted attorneys, as they are not used for training the model (Yang, Lewis, and Frieder, 2021).

Chapter 3

Related work

In this chapter, we discuss the related works in which Technology Assisted Review and semi-supervised learning are used for text classification. First, the autostop framework is discussed (Section 3.1), followed by ASReview (Section 3.2) and FAST^2 (Section 3.3). Lastly, a comparative analysis of semi-supervised learning for relevant article selection is discussed (Section 3.4).

3.1 Autostop

Autostop aims to find the stopping point of TAR. This is done by training a ranking model to rank all the documents and by conducting a sampling method to estimate the total number of relevant documents in the dataset. The total number of relevant documents can be used to calculate the level of recall transparently (see Section 5.3 and Table 5.3 for more information about recall). If the goal is to achieve a certain level of recall, e.g., 95%, then the estimated level of recall can be used to know when to stop the TAR process. The Autostop framework consists of a ranking module, a sampling module, an assessment module, an estimation module, and a stopping module. The framework's output is an estimator of the total number of relevant documents and an estimator of the estimator's variance (Li and Kanoulas, 2020).

The main difference between CAL and Autostop is that Autostop allows random sampling from all documents instead of only sampling the most relevant documents. This random sampling is done with-replacement sampling design to achieve an unbiased estimation of the number of relevant documents with low variance (Li and Kanoulas, 2020).

Autostop proposes two stopping strategies: the first is an optimistic stopping strategy, which stops the TAR process when more relevant documents have been collected than the estimated target number. The second strategy is the conservative strategy. This strategy stops when a higher confidence than the target recall is reached. The difference is that the estimated variance should also be reached on top of the optimistic strategy before stopping. When it is essential that no relevant documents are missed, and the costs are not the primary concern, then the conservative strategy is recommended (Li and Kanoulas, 2020).

Autostop uses TF-IDF, logistic regression, and a ranking model trained topic-wise, which means that the ranking model needs to be trained from scratch for each topic (Li and Kanoulas, 2020).

3.1.1 Knee, Target and Budget Methods

In practice, it can be problematic to measure recall. This is because it is challenging to specify an absolute threshold that should be considered high (Cormack and Grossman, 2015). Also, in practice, the total number of relevant documents in the dataset is not known. Due to this, it is difficult to stop at a certain level of recall in practice.

Three methods are used to find the moment to stop the TAR process. These three methods are the Knee, Target, and Budget methods. These methods are inspired by the gain curve. The gain curve has recall as a function of the number of documents reviewed. This gain curve shows diminishing returns at a certain point. This means that after finding a certain number of relevant documents with high precision, the precision starts to drop, showing that the majority of relevant documents are likely to be found (Li and Kanoulas, 2020).

The Knee method uses geometric algorithms to find a knee in the gain curve. When this knee is found, the TAR process is stopped after a certain amount of diminishing results has occurred compared to before the knee. The Target method reviews a randomly sampled set of documents until a pre-specified number of relevant documents are found. This number is set as the target beforehand. Then, AutoTAR continues to review documents until all the relevant documents from the target set are found. The budget method combines the knee and target methods (Li and Kanoulas, 2020).

AutoTAR uses the Knee method for automatic stopping. No extra assessment costs are needed for the Knee method. However, this method does not provide insights into how many relevant documents are missed by stopping the TAR process. The findings of Li and Kanoulas (2020) show that there is a trade-off of 15% recall to learn how many relevant documents are missed through Autostop. (Li and Kanoulas, 2020).

3.1.2 Scalability of Continuous Active Learning

Scalability of Continuous Active Learning (SCAL) is designed to achieve high recall for large to infinite document collections. This is achieved by using a large fixed-size sample of the dataset. This fixed-size sample of the dataset is used to generate the ranker to estimate the prevalence and to determine the cutoff for a particular target recall. SCAL saves human effort by not exponentially increasing the batch size that needs to be reviewed. Only a stratified sample of the dataset needs to be reviewed (Li and Kanoulas, 2020).

3.1.3 Evaluation metrics

Autostop uses the following evaluation metrics: recall, cost, relative error (RE), and $Loss_{er}$. Recall $= \frac{r}{R}$, where r is the number of relevant documents found, and R is the total number of relevant documents. Cost $= \frac{n}{N}$, where n is the number of documents shown to a human reviewer, and N is the total number of documents. Relative error (RE) is the difference between achieved and target recall. $Loss_{er}$ is used in information retrieval (IR) and combines the recall and cost metrics. $Loss_{er} = (100\% - recall_c)^2 + (\frac{100}{N})^2(\frac{n}{R+100})^2$. Here, $recall_c$ is the achieved recall when TAR is stopped. So, $100\% - recall_c$ represents the loss based on the inability to find all relevant documents. The 100% represents the goal to find 100% of the documents. $\frac{n}{R}$ is the total number of sampled documents divided by the total number of relevant documents. This represents the effort spent in assessing the relevant and non-relevant documents. All these evaluation metrics can be calculated using the open source script tar_eval from `https://github.com/CLEF-TAR/tar` (Li and Kanoulas, 2020).

3.1.4 Datasets

Conference and Labs of the Evaluation Forum (CLEF) Technology-Assisted Reviews in Empirical Medicine datasets, Text Retrieval Conference (TREC) Total Recall datasets, and TREC legal datasets are used in the paper of Li and Kanoulas (2020).

The term prevalence defines the percentage of relevant documents in a dataset. For these datasets, the prevalence is low. It ranges from 0.10% to 2.15% (Li and Kanoulas, 2020)

3.1.5 Results of Autostop

The knee method performs best when the methods are compared to each other based on achieving high recall and low costs. However, the Autostop method performs better when the goal is to stop the TAR process on time. Besides the knee method, Autostop also performed better than the other baselines in terms of high recall and low cost. However, apart from the high recall and low cost, Autostop also provides a transparent, accurate, and effective stopping point, which the other methods do not (Li and Kanoulas, 2020).

A downside of Autostop is that the framework is designed for small-scale datasets. To deal with the larger datasets, Li and Kanoulas (2020) adapted the framework by randomly splitting the datasets and running the algorithms on the split data. Afterward, the sampled documents are concatenated for final review. However, this is not the optimal solution (Li and Kanoulas, 2020).

3.2 ASReview

ASReview aims to help scholars and practitioners get an overview of the most relevant works (documents) as efficiently as possible. This is done by prioritizing relevant studies via active learning. The aim is to balance recall and precision, meaning that the aim is to find as many relevant documents as possible while limiting the number of documents retrieved. Through CAL, the goal is also to minimize the number of labeling tasks by a human. The algorithm aims to find the most relevant documents instead of finding the most accurate model (Schoot et al., 2021).

ASReview aims to be transparent in this process, which is the reason ASReview is an open-source project. ASReview also implemented a benchmark mode, which helps in comparing different algorithms and helps to understand real-world performance. ASReview focuses on systematic reviews, but according to the creators, ASReview can be used for any text source (Schoot et al., 2021).

3.2.1 Features of ASReview

The process of ASReview starts with at least one document labeled in both the relevant and non-relevant classes. Starting with more labeled documents can lead to improved efficiency of the active learning process. In the active learning process, the system shows one document to the human users to be labeled. After labeling, this document is used in training a new model. Then the cycle repeats until a user-defined stopping criterion is met. The default settings for ASReview are a naive Bayes classifier, TF-IDF feature extraction, dynamic resampling balance strategy, and certainty-based sampling for the query strategy. These settings are chosen because they achieve consistently high performance and low computation time. The documents are converted into a document-term matrix, and terms are converted to lowercase. ASReview does not remove stop words by default. Users can change these settings and add new classifiers, feature extraction techniques, query strategies, and balance strategies. More information about the different selectable settings can be found in Table 2 of Schoot et al. (2021).

3.2.2 Balance strategies

The data is usually unbalanced, as is the case with TAR. In a systematic review, most documents are excluded as non-relevant in the title and abstract phases. In this phase, only the titles and abstracts are screened on relevance. A recent study excluded 9,847 documents out of 10,115 in the title and abstract phase, meaning that roughly 97.4% was excluded as non-relevant in this phase.

ASReview has implemented three balance strategies to rebalance the training data: full sampling, undersampling, and dynamic resampling. Full sampling uses all the labeled documents, undersampling excludes a part of the non-relevant class to balance the classes, and dynamic resampling is similar to undersampling. However, dynamic resampling also increases the relevant class size by duplicating documents (Schoot et al., 2021).

Previous research showed that with skewed datasets, resampling methods dealing with class imbalance, such as under-sampling, can significantly improve the performance of machine learning classifiers (Liu, Timsina, and El-Gayar, 2018).

3.2.3 Evaluation metrics

ASReview uses three evaluation metrics. The first two evaluation metrics are based on work saved over sampling (WSS). This measures the percentage of reduction in the number of documents a human has to read compared to a random sampling method (see Section 5.3.1 for more in-depth information about WSS). The first evaluation metric used is WSS@95%, meaning that the work saved is measured at a recall of 95%. WSS@100% is the second evaluation metric used. The third evaluation metric is RRF10%, which is an evaluation metric proposed by Schoot et al. (2021). This metric shows the number of relevant documents found after having screened 10% of the documents (Schoot et al., 2021).

3.2.4 Results of ASReview

On average, ASReview achieved a WSS@95% of 83%. This means that, on average, 95% of the relevant documents were found after screening 17% of the documents. The RRF10% ranged between 70% to 100% (Schoot et al., 2021).

3.2.5 Datasets of ASReview

ASReview has constructed multiple publicly available datasets, which can be found on `https://github.com/asreview/systematic-review-datasets` (Schoot et al., 2021). See Section 5.1 and Table 5.1 for more information on these datasets.

3.2.6 Other related works

A few other related works are abstrackr (using an SVM classifier), Colandr (using an SVM with stochastic gradient descent learning classifier), FASTREAD (using an SVM classifier), Rayyan (using an SVM classifier), RobotAnalyst (using an SVM classifier), which use a variety of inputs, feature extraction, and label options. More information about these related works can be found in Schoot et al. (2021).

3.3 FAST^2

FAST^2 has a similar goal as TAR. FAST^2 aims to optimize and reduce the human effort to find relevant papers. Yu and Menzies (2019) tried to optimize finding the 95% most relevant engineering papers. In their approach, they focused on three key innovations, namely (1) a way to apply external domain knowledge, (2) a way to estimate the number of remaining relevant papers, and (3) an algorithm to correct human error. A full reproduction package is also available at `https://zenodo.org/record/1184123`.

The approaches to find the relevant documents in the FAST^2 domain can be divided into three approaches, namely: (1) search-query-based methods, (2) reference-based methods similar to snowballing, and (3) abstract-based methods (Yu and Menzies, 2019). FAST^2 focuses mainly on abstract-based methods.

3.4 Comparative analysis of semi-supervised learning for relevant article selection

Liu, Timsina, and El-Gayar (2018) compared different semi-supervised learning methods for relevant document selection to automate the process in systematic reviews. Their research goal is to apply semi-supervised learning (SSL) to overcome the labeling bottleneck, i.e., the little availability of labeled data. The process of systematic review consists of three steps. The first step is a keyword search to identify potentially relevant articles. Step two is to identify the articles that need to be included (article triage). The third and final step is to summarize the selected articles.

Article triage (step two of the systematic review) consists of two steps. The first step is called abstract triage. During this step, only the title and abstract of articles are considered during the review to decide whether the articles are relevant or not. In the next step, named full-text triage, the relevant articles from abstract triage are fully inspected to determine whether these articles should be included in the systematic review or not (Liu, Timsina, and El-Gayar, 2018).

In the article triage, supervised learning has been used in almost all research to classify the documents. However, supervised learning with small labeled datasets often leads to overly simple prediction functions. The study of Liu, Timsina, and El-Gayar (2018) is one of the first researches that compare and analyze SSL methods to address small-sized datasets for classification algorithms in medical systematic review creation (Liu, Timsina, and El-Gayar, 2018).

3.4.1 Data and class imbalance

The classes within systematic review are skewed, as is the case with TAR. In the systematic review of the U.S. preventive services task force in 2013, 16,179 articles were found in the keyword research. From this set of articles, 1,190 articles were selected as relevant during the abstract triage. In the final phase, the full-text triage, only 253 articles were included in the systematic review. This is 7.36% relevant articles found in the abstract triage and 1.56% after full-text triage. Such a systematic review requires a significant investment in time and funds, namely 1139 hours of an expert and 250,000 dollars. Also, on average, a systematic review takes 2.4 years. The bottleneck is abstract triage, where scientists screen titles and abstracts of thousands of articles (Liu, Timsina, and El-Gayar, 2018).

A dataset is considered imbalanced when the classes are not equally represented. As only 1.56% of the articles were used after full-text triage, this dataset is imbalanced. As stated before, undersampling can deal with imbalanced data by sampling fewer data from the larger category. This approach can discard valuable data points, and with highly imbalanced data, it may even lead to a lack of data. Oversampling, on the other hand, replicates data points in the minority class. The downside of this approach is that it only increased the weight of random data points in the minority class. Therefore, Synthetic Minority Oversampling Technique (SMOTE) has been used to resolve class imbalance issues that are ubiquitous for medical review datasets. SMOTE aims to overcome overfitting by oversampling. This is done by creating new minority class data points through interpolation between data points of the same class in the neighborhood. This way, it introduces synthetic examples instead of replications (Chawla et al., 2002; Fernandez et al., 2018; Liu, Timsina, and El-Gayar, 2018).

A Unified Medical Language System (UMLS) was used to help boost classification performance when bag-of-words (BOW) is used. However, UMLS is used to extract medical terms. Therefore, it is a technique specific to medical terms and not directly applicable to TAR. Liu, Timsina, and El-Gayar (2018) used the term frequency-inverse document frequency (TF-IDF) to assign weights to each UMLS term. The TF-IDF increases a term (t) when it appears more in a document. However, when it appears in many documents, then it is decreased. More information about TF-IDF can be found in Section 5.2 and Table 5.2.

3.4.2 Semi-supervised learning methods

Liu, Timsina, and El-Gayar (2018) investigated and compared the following semi-supervised learning methods: label spreading, label propagation, and semi-supervised support vector machine (S3VM). Label spreading and label propagation are both graph-based SSL techniques. With label spreading, the labeled data points are used to label the neighboring unlabeled data points based on their proximity. The difference between label spreading and label propagation is that label propagation uses a raw similarity matrix. Through label propagation, the data itself is not modified. However, through label spreading, each iteration uses the modified versions of the graph. This normalized the edge weights by computing the normalized graph Laplacian. S3VM is an extension of support vector machine (SVM), which is a supervised learning technique. S3VM tries to label the unlabeled data so that a linear boundary can be found with the maximum margin on both the labeled data and the unlabeled data that has been labeled in the process. This function is non-convex, which makes it harder to optimize. More information about these SSL methods can be found in Section 4.

3.4.3 Wrapper methods

Apart from these SSL techniques, Liu, Timsina, and El-Gayar (2018) also compared two wrapper methods: self-training and active learning. In short, self-training uses a classifier trained on the labeled data to label the unlabeled data. Usually, this happens in batches. First, the most confident unlabeled points are labeled. Then the classifier is re-trained. After this, the procedure is repeated until no unlabeled data points remain. See Section 4.1.1 for more information on self-training.

Like self-training, active learning is also an iterative process where the newly labeled instances are added to the training set on which it is re-trained.

The following evaluation metrics were used: recall, precision, and F_1. More information about these evaluation metrics can be found in Section 5.3.

3.4.4 Results of SSL methods

The research of Liu, Timsina, and El-Gayar (2018) showed lower results for SVM and S3VM compared to the two graph-based SSL methods. S3VM did achieve a higher recall score than SVM, but S3VM scored lower on precision and F_1. Label spreading and label propagation found more relevant articles than S3VM and SVM. However, label propagation and label spreading also found more False Positives, leading to a lower precision score. As recall is essential, they chose to proceed with the graph-based methods. Label spreading performed better than label propagation for both precision and recall. Therefore, label spreading was selected as the SSL method.

3.4.5 Results of wrapper methods

When experimenting with the self-training in combination with label spreading, they performed 9 to 18 iterations of selecting the top and bottom 8 data points, which for one dataset added 40.44% of the data points to the training set, increasing it from 5% to 45.44%. On this final training set, they trained a supervised SVM to classify the remaining unlabeled data instances, combining both supervised and unsupervised learning. Through this, they significantly increased the precision while still achieving comparable recall scores. This worked exceptionally well on datasets with a small number of seeds (Liu, Timsina, and El-Gayar, 2018).

Active learning resulted in higher recall and precision than self-training and supervised SVM. This could be explained as roughly an equal number of relevant and non-relevant articles being added in each iteration (this was not the case with SVM). Active learning helps identify more relevant articles and achieve a higher recall than supervised learning with random samples (Liu, Timsina, and El-Gayar, 2018).

Chapter 4

Semi-Supervised Learning strategies

Semi-supervised learning (SSL) uses unlabeled data to create better models. SSL can do this by exploiting the marginal distribution of unlabeled data (Søgaard, 2013). Within data mining, SSL has received more attention in the past years due to the potential to reduce the effort to label the unlabeled data (Liu, Timsina, and El-Gayar, 2018).

In SSL, there are two different classification goals. The first goal is to use the current labeled and unlabeled data to predict the labels of future test data. This is called inductive semi-supervised learning. The second goal is to predict the labels of the unlabeled instances in the dataset. This is called transductive learning (Zhu and Goldberg, 2009). For TAR, the latter is most relevant, as our goal is to find the relevant documents in a set of unlabeled documents.

Each semi-supervised model has assumptions about the link between the marginal distribution $p(x)$ and the conditional distribution $p(y|x)$. Due to these different assumptions, choosing the correct semi-supervised learning method for a specific task is essential. Otherwise, including the unlabeled data can decrease the performance compared to a supervised learning method (Zhu and Goldberg, 2009).

In this chapter, we discuss the different semi-supervised learning methods to find the SSL methods suitable for TAR. The SSL methods are grouped into wrapper methods (Section 4.1), graph-based models (Section 4.2), mixture models, and Expectation Maximization (Section 4.3) and lastly semi-supervised support vector machines (Section 4.4). See Table 4.1 for an overview of the different SSL methods that will be discussed.

4.1 Wrapper methods

The wrapper methods discussed in this section can be applied to any existing supervised learning method. Due to this, the choice of which learner we apply is entirely open. The wrapper method wraps around the chosen supervised learner and does not change the inner workings of the supervised learner. The supervised learning method changes into a semi-supervised learning method by applying the

Wrapper methods	**Section**
Self-training	4.1.1
Co-training	4.1.2
Tri-training	4.1.3
Soft self-training (EM)	4.1.4
Multiview learning	4.1.5
Cluster-as-features	4.1.6

Graph based models	**Section**
Label propagation	4.2.1
Semi-supervised nearest neighbor editing	4.2.2
Semi-supervised condensed nearest neighbor	4.2.3
Mincut	4.2.4
Harmonic function	4.2.5

Mixture models and Expectation Maximization	**Section**
Hidden Markov Models	4.3.1
Cluster-then-label methods	4.3.2
Semi-supervised Multinomial Naive Bayes	4.3.3
Expectation Maximization with the many-to-one correspondence	4.3.4
Expectation Maximization with deterministic annealing	4.3.5

Semi-supervised support vector machines (S3VM)	**Section**
Transductive support vector machines	4.4.2
Laplacian support vector machines	4.4.3
meanS3VM	4.4.4
S3VM based on cluster kernels	4.4.5

TABLE 4.1: Overview of the different semi-supervised learning strategies.

wrapper method (Zhu and Goldberg, 2009; Søgaard, 2013). This section will discuss self-training, co-training, tri-training, soft self-training, multiview learning, and cluster-as-features.

4.1.1 Self-training

The self-training wrapper uses the chosen supervised classifier trained on the labeled data to label the unlabeled data. The main downside of self-training occurs when the supervised classifier mislabels some data. The common way to combat this downside is by implementing the strategy to only label an unlabeled data instance when the classifier's confidence is greater than 90% (Søgaard, 2013).

There are multiple ways to make self-training more robust. It is possible to implement throttling, ensuring that only k data points will be selected per pass over the unlabeled data. Balancing ensures that only the k most confidently unlabeled data points are selected per class. Lastly, when pooling is implemented, the unlabeled data selected per pass is a randomly selected subset of the unlabeled data.

Pooling often leads to the best results when only a single data point is pooled per pass over the unlabeled data. However, this is very time-consuming (Søgaard, 2013).

4.1.2 Co-training

Co-training was introduced as another attempt to make self-training more robust. It provides an alternative and second perspective to the unlabeled data (Søgaard, 2013). Co-training represents each data point through two views. The two views can be two (different) sets of features. This is notated as $x = [x^{(1)}, x^{(2)}]$. Co-training is similar to self-training, with the main difference being that co-training uses two different classifiers. The first view teaches the second view and vice versa. This happens when the k most confident predictions over the unlabeled data from the first classifier are added to the second classifier, and again, vice versa (Zhu and Goldberg, 2009).

It is common to split the features randomly between the two views. When random co-training is implemented, a random view is selected when labeling an unlabeled data point (Søgaard, 2013).

The creators of co-training, Blum and Mitchell, proved that co-training could be successful when both views are sufficient, redundant, and conditionally independent of each other. However, later research showed that co-training can still be guaranteed to be successful with a weak dependence. Co-training can work even with sufficiently diverse classifiers trained on the same view. It is possible to use different learning algorithms as different views or two different samples of the labeled data (Søgaard, 2013).

4.1.3 Tri-training

Tri-training utilizes the same concept as co-training. However, now three learners inform each other. Majority voting is applied when an unlabeled data point is labeled by the three learners (Søgaard, 2013).

Generally, tri-training is more robust than self-training. Tri-training also improves significantly over the supervised baseline when less unlabeled data is used. Søgaard (2013) only used the 500 features that correlated best with class according to a χ^2 test.

4.1.4 Soft self-training

With soft self-training, we not only add a label to the unlabeled data points that we are confident about, but we also assign weights to the newly labeled data based on the confidence. Through delible soft self-training (also called generalization of Expectation Maximization), we can run over all unlabeled data each round and refine the weights to the labels (Søgaard, 2013).

4.1.5 Multiview learning

Multiview training is a generalization of co-training. With multiview learning, it is assumed that the algorithm has k different learners. Each learner can be of a different type. It is possible to give each learner access to the whole set of features or use only a subset of them. When the learners are of different types but have access to the same features, it will be similar to the ensemble method. The goal is to minimize its own empirical risk but also agree with all other hypotheses (Zhu and Goldberg, 2009).

4.1.6 Cluster-as-features

The cluster-as-feature, like the wrapper methods discussed above, can be applied to any supervised learning algorithm. Through cluster-as-features, a clustering algorithm learns a clustering model from either the unlabeled data points or from both labeled and unlabeled points. Then, every labeled data point is augmented with a variable that takes $m_U(x_n)$. This new variable encodes the marginal distribution of the unlabeled data and may enable us to better generalize beyond our labeled sample (Søgaard, 2013).

4.2 Graph-based models

In graph-based models, the distance between two data points is measured. The edge weight is typically large when the two data points are close to each other. The model assumes that when two data points are connected with a large weight, the two data points tend to have the same label (Zhu and Goldberg, 2009).

Usually, the graphs are undirected. The usual edge weights used for graph models are fully connected graphs with Euclidean distance, k-Nearest Neighbor graphs based on Euclidean distance, and ϵ Nearest Neighbor graphs, where the edges will connect if the Euclidean distance is smaller than ϵ (Zhu and Goldberg, 2009).

The semi-supervised learning algorithms in the first 3 subsections below, label propagation, semi-supervised nearest neighbor editing, and semi-supervised condensed nearest neighbor are all tied to the supervised algorithm nearest neighbor (Søgaard, 2013). After those subsections, Mincut and harmonic function will be discussed, which are both transductive learning algorithms.

4.2.1 Label propagation

Label propagation is one of the earliest graph-based semi-supervised algorithms. It uses a k-nearest neighbor kernel. This makes it similar to self-training with k-nearest neighbor when using weighted voting. With label propagation, the edges are weighted so that when two vertices are close to each other, their edge weight is high. The distance between two vertices is calculated with the Euclidean distance (E) and a heuristic (σ). The weight (w) between vertice x_i and x_j is

$$w_{ij} = exp\frac{-E(x_i, xj)^2}{\sigma^2}.$$

In each iteration, all nodes collect votes on which class they belong. This includes their own label if it is a labeled data point (Søgaard, 2013).

4.2.2 Semi-supervised nearest neighbor editing

As with support vector machines (SVM), many data points can be disregarded without affecting the classification performance. Nearest neighbor editing is an outlier detection technique that can be used to make the nearest neighbor methods more efficient. This is done by disregarding all the data points that their k-nearest neighbors can predict. This technique tries to improve the speed, not the performance. This is useful since a brute-force nearest neighbor search will take a long time on large data sets since it runs time linear in the size of the training set (Søgaard, 2013).

The downside of semi-supervised nearest neighbor editing is that performance can dramatically drop when the data is biased. This is the case because outliers are often essential to classify instances from new domains (Søgaard, 2013).

4.2.3 Semi-supervised condensed nearest neighbor

The condensed nearest neighbor (CNN) aims to return a single point in the center of each cluster. The problem that can occur with CNN is that the labeled data does not necessarily include data points near the center of each cluster. Semi-supervised condensed nearest neighbor (SCNN) labels the unlabeled data points, which increases the chance of including data points near the cluster center. The unlabeled data is labeled through the *k*-nearest neighbor classifier on the original dataset. Only the cases with a confidence greater than 90% are labeled (Søgaard, 2013).

Søgaard (2013) discusses two papers that show that CNN performs roughly the same as nearest neighbor. However, the SCNN sometimes leads to substantial improvements over the *k*-nearest neighbor methods.

4.2.4 Mincut

Both Mincut and the harmonic function are transductive learning algorithms. In Mincut, positively labeled vertices are called source vertices, and negatively labeled vertices are called sink vertices. The objective of Mincut is to find the minimum set of edges that need to be removed to block all flows from source nodes to sink nodes. After splitting the graph, the vertices that are connected to source nodes are labeled positive, and the vertices connected to the sinks are labeled negative (Zhu and Goldberg, 2009).

A flaw of the formulation of Mincut is that it is possible to have multiple equally good solutions. Due to this, some vertices can be positive or negative based on which solution is chosen (Zhu and Goldberg, 2009).

4.2.5 Harmonic function

In harmonic function, the labeled vertices have the same values as their label. The unlabeled vertices receive the value of the weighted average of their neighbors' values. This satisfies the weighted average property on the unlabeled data. Now, the function f can produce real values. However, this can be addressed by applying a threshold at $f(x) = 0$. So, when $f(x) \geq 0$, then we assign label 1, and when $f(x) < 0$, we assign label 0. The function f has a closed-form solution that is unique (under mild conditions) and is globally optimal (Zhu and Goldberg, 2009).

The harmonic function is computed in an iterative procedure. The first step is to set $f(x_i) - y_i$ for all the labeled vertices. The unlabeled vertices start with an arbitrary value. In each iteration, the labels of the unlabeled vertices are calculated by the weighted average of the neighbors. This is done through

$$f(x_i) = \frac{\sum_{j=1}^{l+u} w_{ij} f(x_j)}{\sum_{j=1}^{l+u} w_{ij}}.$$

Regardless of the starting values of the unlabeled vertices, this procedure is guaranteed to converge to the harmonic function. This iterative procedure is also called label propagation, since it propagates labels from the fixed labeled vertices to the unlabeled vertices (Zhu and Goldberg, 2009).

4.3 Mixture models and Expectation Maximization

With mixture models, the aim is to use unlabeled data to learn how the mixed instances from the different classes are distributed. After the model has learned how each separate class is distributed, we may be able to decompose the mixture into individual classes (Zhu and Goldberg, 2009).

This is done by selecting the parameters that maximize the probability of generating the training data through the proposed model. The $p(y|x)$ can be computed through the generative model, which uses the Bayes rule:

$$p(y|x) = \frac{p(x|y)p(y)}{\sum_{y'} p(x|y')p(y')}.$$

This formula contains the class conditional distributions and the prior probabilities (Zhu and Goldberg, 2009).

It is possible to use the maximum likelihood estimate (MLE) for supervised methods. This is no longer possible with semi-supervised learning methods. However, when working with semi-supervised learning, it is possible to find a local maximum through the EM algorithm (Zhu and Goldberg, 2009).

In the subsections below, a short description of Hidden Markov models, and cluster-then-label models will be given, followed by semi-supervised Multinomial Naive Bayes which uses Expectation Maximization (EM).

4.3.1 Hidden Markov Models (HMM)

The Hidden Markov model is a generative model. However, this model is commonly used to model sequences of data (Zhu and Goldberg, 2009). Text is inherently sequential. However, this is ignored in the BOW representation. As the BOW representation is used in this research, we will not look further into Hidden Markov Models.

4.3.2 Cluster-then-label methods

Unsupervised clustering algorithms can identify clusters from unlabeled data. Through this, it is possible to cluster the data and then label the unlabeled data for semi-supervised classification. This method does not necessarily involve probabilistic mixture models. Different types of linkage can be used. When single linkage is used, the clusters become longer and more skinny. With complete clustering, the clusters tend to be rounder (Zhu and Goldberg, 2009).

4.3.3 Semi-supervised Multinomial Naive Bayes

Assumptions

The theoretical basis of Expectation-Maximization (EM) shows that it is possible to find a more probable model when there is a large enough set of unlabeled data. A more probable model will also result in a more accurate classifier. However, this depends on whether the assumptions of the generative model are correct.

There are three assumptions with a probabilistic generative model with a naive Bayes classifier: (1) The data are produced by a mixture model, (2) there is a one-to-one correspondence between mixture components and classes, and (3) the mixture components are multinomial distributions of individual words (Nigam, McCallum, and Mitchell, 2006).

Representation of the documents

The commonly used naive Bayes represents each document as a bag of words (BOW). BOW is a simple document representation, as it disregards all word ordering information (Nigam, McCallum, and Mitchell, 2006).

Nigam, McCallum, and Mitchell (2006) found that when the optimization of the naive Bayes model probability is strongly correlated with the classification accuracy, the naive Bayes model is sufficient for text classification, as EM optimizes on posterior model probability. When there is a correlation between model probability and accuracy, EM can indirectly optimize accuracy. However, when the naive Bayes generative model is not well correlated with the classification accuracy, adopting a more expressive generative model can restore this correlation.

As stated in Section 4.3.3, it is assumed that documents are generated by a mixture of multinomials model, where each mixture component corresponds to a class. Every document is generated according to a probability distribution. The parameters of the probability distribution are denoted as θ. Parameters in the mixture model define this probability distribution. It is assumed that the class label for a document is a one-to-one correspondence with the mixture component. So, when a mixture component (c_j) generates a document, then the class label of this document is also $y_i = c_j$. It is also assumed that the words of a document are conditionally independent of the other words in the document (Nigam, McCallum, and Mitchell, 2006).

Text classification

Naive Bayes commonly uses the maximum a posteriori (MAP) estimate to estimate the parameters in the mixture model. This is the most probable given the evidence of training data and a prior. With the estimates of the parameters based on the labeled documents, we can turn the generative model backward and calculate the probability that a mixture component generated an unlabeled document (x_i). The class of a document is then equal to the class with the highest posterior probability (Nigam, McCallum, and Mitchell, 2006).

As the labels of the unlabeled data points are unavailable, it is impossible to use closed-form equations. Through Expectation Maximization (EM), we can find local MAP parameter estimates for the generative model (Nigam, McCallum, and Mitchell, 2006).

Downside of Expectation Maximization

A downside of EM is that it only guarantees the discovery of local maxima. This is especially an issue within the text classification domain because there are many parameters. To combat this, deterministic annealing and another modeling estimation find more probable and more accurate classifiers (Nigam, McCallum, and Mitchell, 2006).

EM algorithm for semi-supervised Multinomial Naive Bayes

The algorithm of the EM technique starts with building a naive Bayes classifier based on only the labeled documents. This happens in a supervised fashion. Then, all the unlabeled documents are classified by this naive Bayes classifier. While classifying the unlabeled documents, not only the most likely class is noted. Instead, the probabilities associated with each class are noted. First, the word probability estimate per class is calculated through Formula 4.1.

$$\hat{\theta}_{w_t|c_j}^{(t+1)} \equiv p(w_t|c_j;\hat{\theta}^{(t)}) = \frac{1 + \sum_{x_i \in X} \delta_{ij} x_{it}}{|\mathcal{X}| + \sum_{s=1}^{|\mathcal{X}|} \sum_{x_i \in X} \delta_{ij} x_{is}} \tag{4.1}$$

Here, the probability of a word w_t is calculated, given the mixture component c_j and the estimate of the model parameters $\hat{\theta}$. The 1 in the numerator and the $|\mathcal{X}|$ in the denominator, which represents the vocab size, are added for smoothing. In the numerator, for each document $(x_i \in X)$, the number of times the word w_t occurs in document x_i, represented as x_{it}, is summed. This is only done for the documents where the component equals the given c_j. This is done by δ_{ij}, which equals 1 when the documents component is equal to the given c_j, else it is 0. In the denominator, the same summation happens, but now for all words $(\sum_{s=1}^{|\mathcal{X}|})$ (Nigam, McCallum, and Mitchell, 2006).

Formula 4.2 is used to calculate the class probabilities.

$$\hat{\theta}_{c_j}^{(t+1)} \equiv p(c_j|\hat{\theta}^{(t)}) = \frac{1 + \sum_{i=1}^{|X|} \delta_{ij}}{M + |X|} \tag{4.2}$$

The 1 in the numerator and the M in the denominator, which represents the total number of classes, are added for smoothing. In the numerator, the number of documents that belong to component c_j are summed. This is represented through δ_{ij}, which is only equal to 1 when the document class equals the given component c_j. In the denominator, the total number of documents $(|X|)$ are added to the smoothing (Nigam, McCallum, and Mitchell, 2006)

Lastly, Formula 4.3 calculates the probability that a document class label y_i equals the selected component c_j, given the document x_i and the estimate of the model parameters $\hat{\theta}$.

$$\delta_{ij} = p(y_i = c_j|x_i;\hat{\theta}) = \frac{p(c_j|\hat{\theta}) \prod_{w_t \in \mathcal{X}} p(w_t|c_j;\hat{\theta})^{x_{it}}}{\sum_{k=1}^{M} p(c_k|\hat{\theta}) \prod_{w_t \in \mathcal{X}} p(w_t|c_k;\hat{\theta})^{x_{it}}} \tag{4.3}$$

In the numerator, the class probability of c_j (Formula 4.2) is multiplied by the word probability estimate (Formula 4.1) of each word in the vocabulary. x_{it} represents the number of times word w_t occurs in document x_i. In the denominator, the same calculation is made. However, here it is summed over all classes (Nigam, McCallum, and Mitchell, 2006).

Through Formula 4.3, all the unlabeled documents have estimated class probabilities. These estimated class probabilities are used as true class labels and train the naive Bayes classifier based on both labeled and unlabeled data. Unlabeled data is handled as fractional documents similar to their estimated class probabilities. This process is iterated until it converges (Nigam, McCallum, and Mitchell, 2006).

During the E-Step, the unlabeled documents are labeled. A new MAP estimate for the parameters (θ) is calculated in the M-Step. As the parameters lead to one of the local maxima, many instantiations of EM will be chosen. Instead of using a random starting point, it is possible to use the labeled data to select a starting point (Nigam, McCallum, and Mitchell, 2006).

In case the assumptions of the generative model do not hold, then the benefits of unlabeled data become less clear. Naive Bayes tends to produce extreme class probabilities when the word independence assumption does not hold. However, accuracy can still be high, even if these estimates are extremely high or low. However, it is important to note that SSL leans more on the correctness of modeling assumptions than supervised learning (Nigam, McCallum, and Mitchell, 2006).

Results of EM compared to naive Bayes

In an experiment of Nigam, McCallum, and Mitchell (2006), naive Bayes achieved 52% accuracy, while EM achieved 66% accuracy. This was especially the case when labeled data was sparse. When there is already enough labeled data, the unlabeled data does not help as much (it only improved from 76% to 78%). The significantly higher accuracy is achieved by optimizing the posterior model probability instead of directly optimizing the classification accuracy. Generative models are representative enough for text classification when the model's probability and accuracy are correlated. This allows EM to optimize accuracy indirectly. The correlation coefficient was 0.9798, which shows a very strong correlation between the accuracy and model probability (Nigam, McCallum, and Mitchell, 2006).

4.3.4 Expectation Maximization with many-to-one correspondence

Correspondence between mixture component and classes

The assumption of one-to-one correspondence between classes and components in the mixture model is dangerous in text classification and, thereby, in TAR. It will lead to an unrepresentative model when all documents in the non-relevant class are modeled as only a single multinomial distribution. This is because the non-relevant documents can contain documents of a variety of different sub-topics. Therefore, a many-to-one correspondence between the mixture component and classes would increase the representativeness of a model. This replaces the one-to-one assumption with a less restrictive one (Nigam, McCallum, and Mitchell, 2006).

Through this new approach, we miss the class and sub-topic of the unlabeled data. From the labeled data, we only miss the sub-topic (represented by z_i). This changes the way of calculating the class membership of a document. The formula for the word probability is still identical to Formula 4.1.

$$\hat{\theta}^{(t+1)}_{w_t|c_j} \equiv p(w_t|c_j;\hat{\theta}^{(t)}) = \frac{1+\sum_{x_i\in X}\delta_{ij}x_{it}}{|\mathcal{X}|+\sum_{s=1}^{|\mathcal{X}|}\sum_{x_i\in X}\delta_{ij}x_{is}} \tag{4.4}$$

The formula for the class probabilities is comparable to Formula 4.2. The only difference is the notation for classes (t_a) instead of components (c_j).

$$\hat{\theta}_{t_a}^{(t+1)} \equiv p(t_a|\hat{\theta}^{(t)}) = \frac{1 + \sum_{i=1}^{|X|} \delta_{ia}}{M + |X|} \tag{4.5}$$

The calculation for the sub-topic probabilities (Formula 4.6) is comparable to the formula for class probabilities (Formula 4.5).

$$\hat{\theta}_{c_j|t_a}^{(t+1)} \equiv p(c_j|t_a; \hat{\theta}^{(t)}) = \frac{1 + \sum_{i=1}^{|X|} \delta_{ij}\delta_{ia}}{\sum_{j=1}^{N} q_{aj} + \sum_{i=1}^{|X|} \delta_{ia}} \tag{4.6}$$

In the numerator, the term δ_{ij} is now equal to 1 when the sub-topic equals c_j, and δ_{ia} equals 1 when the document belongs to class t_a. So, $\sum_{i=1}^{|X|} \delta_{ij}\delta_{ia}$ represents the number of documents belonging to sub-topic j of class a, as only the documents with the correct sub-topic and class are summed. In the denominator, similar to the class probabilities, the $\sum_{j=1}^{N} q_{aj}$ part represents the number of sub-topics of class a, as q_{aj} is only 1 if sub-topic j belongs to class a. This is added for for smoothing. The $\sum_{i=1}^{|X|} \delta_{ia}$ part represents the number of documents belonging to class a, as δ_{ia} equals 1 when the document belongs to class t_a (Nigam, McCallum, and Mitchell, 2006).

Finally, Formula 4.7 calculates the sub-topic membership of a given document x_i.

$$\delta_{ij} = p(z_i = c_j|x_i; \hat{\theta}) = \frac{\sum_{a\in[M]} q_{aj} p(t_a|\hat{\theta}) p(c_j|t_a; \hat{\theta}) \prod_{w_t\in X} p(w_t|c_j; \hat{\theta})^{x_{it}}}{\sum_{r\in[N]} \sum_{b\in[M]} q_{br} p(t_b|\hat{\theta}) p(c_r|t_b; \hat{\theta}) \prod_{w_t\in X} p(w_t|c_r; \hat{\theta})^{x_{it}}} \tag{4.7}$$

Here $\sum_{a\in[M]} q_{aj}$ sums over each class, but only if the sub-topic c_j belongs to class t_a. $p(t_a|\hat{\theta})$ is the class prior (Formula 4.5). $p(c_j|t_a; \hat{\theta})$ is the sub-topic probability (Formula 4.6). Finally, the product of the word probability (Formula 4.4) for each word w_t is taken in $\prod_{w_t\in X} p(w_t|c_j; \hat{\theta})^{x_{it}}$. The denominator has the same calculation as the nominator, except now it sums over all the sub-topics through $\sum_{r\in[N]}$.

So, through Formula 4.7, the sub-topic membership of a document is calculated. As a final step, Formula 4.8 is used to calculate the overall class membership of a document. This is done by summing the probabilities of all sub-topics for the given class t_a. (Nigam, McCallum, and Mitchell, 2006).

$$\delta_{ia} = p(y_i = t_a|x_i; \hat{\theta}) = \sum_{j\in[N]} q_{aj} p(z_i = c_j|x_i; \hat{\theta}) \tag{4.8}$$

EM with the many-to-one correspondence

As we still do not have all the class and sub-topic labels, we will again use EM. The M-step still builds the maximum a posteriori parameter estimates for the multinomials and priors. The difference is that it now uses the probabilistic class and sub-topic estimates. The E-Step calculates the probabilistically-weighted sub-topic and class membership for the unlabeled documents. For the labeled documents, only the sub-topic is calculated. However, only the sub-topics belonging to the known class need to be considered (Nigam, McCallum, and Mitchell, 2006).

For model selection, cross-validation is used as a limited amount of labeled documents are available. Another plus of using multiple mixture components per class is that the model can now capture some dependencies between words on the class level. One multinomial can now cover two words that co-occur within a sub-topic (Nigam, McCallum, and Mitchell, 2006).

Finding the best number of mixture components

The next step is to find a method to find the best number of mixture components without having access to the labels. The number of sub-topics chosen is part of the tension between the complexity of the model and data sparsity. With a too large number of sub-topics, we can perfectly model the training data. However, the generalizability will suffer due to data sparsity. The multinomials will have many parameters estimated from only a few documents. When too few sub-topics are chosen, we will have accurate estimations of the multinomials. However, the model will not represent the true document distribution. This is due to the too-restrictive model.

Cross-validation can be used to select a good compromise in this tension. This can be done through leave-one-out cross-validation. However, in the experiments of Nigam, McCallum, and Mitchell (2006), it is shown that leave-one-out cross-validation does not choose the best number of components. In their research, leave-one-out cross-validation chose a smaller number of components than was best on the test set.

4.3.5 Expectation Maximization with deterministic annealing

Deterministic annealing

In text classification tasks, convergence is usually fast. This means we can trade some convergence speed for an improved local maxima situation. This brings us to deterministic annealing (Nigam, McCallum, and Mitchell, 2006).

Deterministic annealing starts with a simple surface and changes this step by step to become bumpier and, thereby, closer to the true probability surface. When it starts with the simple surface's original maximum, a highly probable maximum will be found when the surface gets more complex. Through this, many local maxima are avoided. This can be set with a single parameter β. When β is 1, the surface space correlates well to the classification accuracy. However, when β approaches 0, the surface becomes convex with a single global maximum (Nigam, McCallum, and Mitchell, 2006).

The idea is to initialize the search with the old maximum while gradually raising β. When $\beta = 1$, a good local maximum will be found. However, the computational costs of deterministic annealing are significantly higher than EM. An example of Nigam, McCallum, and Mitchell (2006) needed 390 iterations with deterministic annealing while only needing 7 iterations for EM. The iterations of deterministic annealing and EM take the same computation. This makes deterministic annealing take 55.7 times longer to compute than EM in their experiment.

The high-probability models produced through deterministic annealing correspond with high-accuracy classifiers, showing that deterministic annealing makes good use of unlabeled data for text classification (Nigam, McCallum, and Mitchell, 2006).

Results of deterministic annealing

The experiments of Nigam, McCallum, and Mitchell (2006) showed that deterministic annealing indeed could improve classification when the class-to-component correspondence was solved. It finds high-probability models, which are then correlated with high-accuracy classifiers. Humans could trivially solve the class-correspondence problem. Humans could identify a class given the most indicative words in almost all cases. It was a small amount of human effort to correct the class correspondence after deterministic annealing. This could be part of the CAL. Through this, the class labels would be mapped to the cluster components and lead deterministic annealing to be successful in finding more probable and more accurate models than EM alone (Nigam, McCallum, and Mitchell, 2006).

Using EM to fix the correspondence of deterministic annealing

EM should also be able to fix the correspondence of the deterministic annealing model, as EM usually has the correct class correspondence. This can be achieved by training an EM alongside deterministic annealing. The correspondence can be fixed by measuring the distance between the EM class multinomial and the deterministic annealing class multinomial. The matrix of distances can then be used to assign the class labels found in EM to the closest match in the deterministic annealing model (Nigam, McCallum, and Mitchell, 2006).

4.4 Semi-supervised support vector machines (S3VM)

The standard form of SVM is a supervised learning method that can not make good use of unlabeled data points. The strong points of SVM are that it has a good theoretical foundation, it finds the global maximum instead of the local maximum, the sparsity of the solution, it is nonlinear, and it has good generalization. However, when there is sparsity in labeled data, SVM would not be strong in generalization. This is where a semi-supervised learning method can help improve SVM. S3VM is a semi-supervised method that introduces SSL with SVM and, therefore, can make good use of these unlabeled data points (Ding, Zhu, and Zhang, 2017). Previous research concluded that the generalization of a classifier can be improved with a significant increase of unlabeled data (Ding, Zhu, and Zhang, 2017).

In short, SVM tries to find an optimal classification hyperplane. When H is the classification hyperplane, then H_1 and H_2 are planes that go through the closest data points to hyperplane H. These data points are called support, and the distance between H_1 and H_2 is the maximum margin. Both H_1 and H_2 are parallel to H.

Originally, S3VMs were called Transductive Support Vector Machines (TSVMs). S3VMs need another loss function than SVMs. This is because the labels of the unlabeled data points are not known. Therefore, it is not known whether an unlabeled data point is on the right or wrong side of the decision boundary. The loss function used for S3VM is the hat loss function. This loss function penalizes unlabeled instances when $-1 < f(x) < 1$. So, it penalizes instances around $f(x) \approx 0$, i.e., around the decision boundary. S3VM is non-convex and non-probabilistic. It does not compute the label posterior probability $p(y|x)$ (Zhu and Goldberg, 2009).

The goal of S3VM is to utilize both labeled and unlabeled data points to build a classifier. S3VM tries to find the maximum margin when separating the labeled and unlabeled data. The new optimal classification boundary must also satisfy that the classification on original unlabeled data and have the smallest generalization error (Ding, Zhu, and Zhang, 2017).

S3VM's can be used in the case of two-class problems, which is the case within TAR. The performance of the classifier would be better if a semi-supervised SVM were used when there is a small amount of labeled data and a large amount of unlabeled data available. S3VM's were initially used in text classification, where it achieved good results (Ding, Zhu, and Zhang, 2017).

There are two assumptions on which a variety of SSL algorithms are based. The first one is the cluster assumption. This means that when two data points in the same cluster have a higher probability of belonging to the same class. The second assumption is the Manifold assumption. The Manifold assumption entails that two data points close to each other along the manifold flat have a similar nature or similar label. (Ding, Zhu, and Zhang, 2017).

In the subsections below, the assumptions, transductive support vector machines, laplacian support vector machines, meanS3Vm, and S3VM based on cluster kernels will be discussed, followed by a conclusion, where the different S3VMs are compared to each other.

4.4.1 Assumptions

There are two assumptions on which a variety of SSL algorithms are based. The first one is the cluster assumption. This means that when two data points in the same cluster have a higher probability of belonging to the same class. The second assumption is the Manifold assumption. The Manifold assumption entails that two data points close to each other along the manifold flat have a similar nature or similar label. (Ding, Zhu, and Zhang, 2017).

4.4.2 Transductive support vector machines

The transductive support vector machine (TSVM) was proposed in 1999 and is based on the cluster assumption mentioned in section 4.4.1. TSVM focuses on a particular working set. Due to this, it is able to achieve the optimal classification within this working set. However, due to this, TSVM also has poor generalization. TSVM also has high time complexity. Lastly, for TSVM to work, the number of positive labels in the workset need to be appointed before training (Ding, Zhu, and Zhang, 2017).

4.4.3 Laplacian support vector machines

The Laplacian support vector machine (LapSVM) is a graph-based SSL method. The manifold assumption mostly reflects the graph-based SSL methods (see Section 4.4.1). In the graph, the similarities between two data points can be represented by edge weights. In turn, the labeled data points can be used to label the unlabeled data. LapSVM is applied in image classification (Ding, Zhu, and Zhang, 2017).

When LapSVM is compared to S3VM, it is shown that LapSVM is more outstanding in solving and time complexity. LapSVM reflects the local structure information of the dataset, as it is based on the manifold assumption. This assumption can be too broad for some classification problems (Ding, Zhu, and Zhang, 2017).

4.4.4 meanS3VM

The meanS3VM was proposed in 2009. Here, the label mean is used to build the SSL. The label mean is a simple statistic. The goal of meanS3VM is to maximize the margin between the label means of the unlabeled data.

The first step is to estimate the label mean of the unlabeled data set. After making this estimation, meanS3VM is similar to SVM. This makes meanS3VM more efficient. This turned out to be the case, as meanS3VM is 100 times faster in training time than TSVM and 10 times faster than LapSVM with relatively large datasets. (Ding, Zhu, and Zhang, 2017).

4.4.5 S3VM based on cluster kernels

Kernel functions are constructed by using labeled and unlabeled data to improve the kernel function in reflecting the similarity between samples. In 2002, a framework was proposed for constructing kernels that rely on the cluster assumption. This framework allowed for an S3VM based on cluster kernels. The kernel function, which is constructed by using both labeled and unlabeled data, is then used to train the SVM (Ding, Zhu, and Zhang, 2017).

When the objective function of an SVM is adjusted, solving and low efficiency would become more difficult. An advantage of S3VM based on cluster kernels is that it does not modify the objective function of SVM. S3VM based on cluster kernels only needs to create a new kernel function. The downside of this is that it has high time complexity and may have poor performance with large datasets (Ding, Zhu, and Zhang, 2017).

4.4.6 Conclusion

According to Ding, Zhu, and Zhang (2017), lapSVM and meanSVM are the outstanding algorithms. When considering the solving process, lapSVM and S3VM are easier than the others. They are based on the cluster kernel. When comparing time complexity, lapSVM and meanSVM have good performance. However, on large datasets, especially meanSVM had a good performance. When comparing based on accuracy, it becomes more difficult. This is because the algorithms have their promise in different datasets. This prevents an accurate estimation of which algorithm is the best.

The upside of S3VM is that it inherits the solid theory of SVM, which improves the method’s generalization ability. The downside, however, is the high time costs.

Chapter 5

Experimental evaluation setup

In this chapter we discuss the setup of the experimental evaluation. First, the different datasets are discussed. Then, the features and evaluation metrics are discussed. After that, the implementation of the four semi-supervised models is discussed. Finally, the parameter tuning and experiment setup is discussed.

5.1 Datasets

The datasets provided by ASReview are used for this research. These datasets are publicly available at `https://github.com/asreview/systematic-review-datasets`. From the 27 available datasets, 12 were chosen that were already usable within the framework from the National Police Lab AI (Bron, 2021b). The framework's existing dataset import function cannot import the other datasets yet. The dataset sizes of the 12 chosen datasets vary between 1704 and 10953 documents, and the number of relevant documents varies between 11 and 280. An overview of these datasets can be found in Table 5.1.

The datasets of ASReview have already been preprocessed. The datasets are retrieved from the Open Science frame (OSF) or Zenodo and the documents are already labeled as either relevant or irrelevant (ASReview, 2022).

Name	**Topic**	**Size**	**Rel #**	**Rel %**
Appenzeller-Herzog_2020	Wilson disease	3453	29	0.84%
Bannach-Brown_2019	Animal Model of Depression	1993	280	14.05%
Bos_2018	Dementia	5746	11	0.19%
Hall_2012	Software Fault Prediction	8911	104	1.17%
Kitchenham_2010	Software Engineering	1704	45	2.64%
Kwok_2020	Virus Metagenomics	2481	120	4.84%
Nagtegaal_2019	Nudging	2019	101	5.00%
Radjenovic_2013	Software Fault Prediction	6000	48	0.80%
Wahono_2015	Software Defect Detection	7002	62	0.89%
Wolters_2018	Dementia	5019	19	0.38%
van_Dis_2020	Anxiety-Related Disorders	10953	73	0.67%
van_de_Schoot_2017	PTSD Trajectories	6189	43	0.69%

TABLE 5.1: Overview of the 12 ASReview datasets

5.2 Document representation

Documents are represented as term frequency-inverse document frequency (TF-IDF) vectors. TF(t, d) stands for Term Frequency and is calculated by dividing the number of times a term (t) appears in document (d) by the total number of terms in document (d). IDF stands for Inverse Document Frequency and is calculated by dividing the total number of documents by the number of documents containing the term (t). TF-IDF is TF(t, d) multiplied by IDF(t) (Cohen et al., 2006). The TF-IDF score increases for a term (t) when it appears more often in a document. However, when it appears in many documents, then it is decreased. The formulas for TF-IDF can be found in Table 5.2

Metric	Formula
TF(t, d)	$\frac{\text{The number of times a term (t) appears in a document (d)}}{\text{The total number of terms in (d)}}$
IDF(t)	$\frac{\text{The total number of documents}}{\text{The number of documents containing the term (t)}}$
TF-IDF(t, d)	$TF(t,d) \times IDF(t)$

TABLE 5.2: Formulas of the document representation.

5.3 Evaluation metrics

5.3.1 Recall and work saved over sampling (WSS)

We use recall and work saved over sampling (WSS) as evaluation metrics. The formulas of these evaluation metrics are shown in Table 5.3. These evaluation metrics are based on True Positives (TP), False Negatives (FN), and False Positives (FP). TP are data instances correctly classified as positive, in our case, relevant documents. False represents a miss-classification. Therefore, FP are data instances that are miss-classified as positive (relevant), while they were negative (irrelevant documents). FN are instances that are miss-classified as negative (irrelevant documents), while they are positive (relevant documents).

Using this terminology, recall is the proportion of relevant documents that has been classified correctly. So, this represents how many of the relevant documents were found and labeled as relevant by the algorithm.

The main goal of TAR is to maximize recall while minimizing the reviewer's effort (Cormack and Grossman, 2015). So, TAR has to save the human reviewer's work of reading all the documents. We measure the work saved as a percentage of documents the reviewer does not have to read due to the use of TAR. This percentage needs to be higher than a method that uses random sampling. Therefore, work saved over sampling (WSS) measures the work saved compared to random sampling, given a required level of recall (Cohen et al., 2006).

The WSS score is calculated by first summing the True Negatives (TN) with the False Negatives (FN). This represents the number of documents predicted to be irrelevant and, thereby, the number of documents the reviewer did not have to read. The $(TN + FN)$ value is then divided by the total number of documents (N) to make it a proportion. Achieving a recall of 95% through a random sampling method would, on average, result in the reviewer not having to read 5% of the documents. Therefore, if the required level of recall is 95%, this 5% is deducted from the percentage of documents the reviewer did not have to read. This is represented in the formula as $-(1.0 - R)$. This way, the WSS score represents the percentage of documents the reviewer does not have to read compared to a random sampling method. See Table 5.3 for the WSS formula.

Metric	Formula	Description
Recall	$\frac{\text{TP}}{\text{TP+FN}}$	Proportion of correctly classified relevant documents compared to all relevant documents.
WSS	$\frac{\text{TN + FN}}{N} - (1.0 - R)$	Work saved compared to simple sampling.
WSS@95%	$\frac{\text{TN + FN}}{N} - 0.05$	Work saved compared to simple sampling at the moment when the recall score is 95%.

TABLE 5.3: Formulas of the evaluation metrics. TP stands for True Positives, TN stands for True Negative, FP for False Positives, FN for False Negatives, R stands for the required recall value and N stands for the total number of documents.

5.3.2 Exclusion of evaluation metrics

This research excluded accuracy and Area under the ROC curve (AUC) as evaluation metrics. This is because evaluation metrics based on accuracy, i.e., $\frac{\text{TP+TN}}{\text{TP+FP+FP+FN}}$ do not work properly when there is a class imbalance. As discussed in Section 3.2.2, this is the case with TAR. For example, when there are only 0.19% relevant documents (as with the van_Dis_2020 dataset), 99.81% accuracy can be achieved by simply labeling all documents as non-relevant. Accuracy also assumes an equal misclassification cost (Liu, Timsina, and El-Gayar, 2018). This is not the case with TAR, as it is more costly to have a relevant document labeled as irrelevant than an irrelevant document classified as relevant. In the first case, the relevant document will never be shown to a human reviewer, meaning that the document will not be found. In contrast, the second case means that the human reviewer has to read a non-relevant document, which is only costly in time. This is why recall is an important evaluation metric in this research (Liu, Timsina, and El-Gayar, 2018).

5.4 Model algorithms

This research will compare six semi-supervised learning models to their supervised learning counterparts. The models used in this thesis are Labelspreading, semi-supervised Multinomial Naive Bayes, semi-supervised Multinomial Naive Bayes with sub-topics, SVM with self-training, Logistic Regression (used in AutoTAR), Multinomial Naive Bayes, SVM, and two models where Logistic Regression is combined with semi-supervised Multinomial Naive Bayes with sub-topics, which we will call CombiTAR.

The framework of the National Police Lab AI (Bron, 2021b) already uses supervised learning methods in its TAR implementation. The implemented supervised models are models imported from scikit-learn. Since the framework of the National Police Lab AI already has functionalities implemented to work with scikit-learn models, mainly scikit-learn models are used.

5.4.1 The semi-supervised learning models

Label spreading

For label spreading, the scikit-learn model is imported (Pedregosa et al., 2011).

Multinomial Naive Bayes (with sub-topics)

The semi-supervised Multinomial Naive Bayes with sub-topics model is based on the paper of Nigam, McCallum, and Mitchell (2006). There was no publicly available implementation of this semi-supervised model. Therefore, the Multinomial Naive Bayes with sub-topics model is built based on the paper. Scikit-learn did also not have a comparable semi-supervised learning model available, as they currently only have three semi-supervised learning methods available, namely: Label propagation, Label spreading, and Self-training (Scikit-learn, 2023a).

As the basis of the Multinomial Naive Bayes with sub-topics model, the scikit-learn model Multinomial Naive Bayes is used (Pedregosa et al., 2011). Adjustments are made to both the fit, predict, and predict_probabilities functions of the Multinomial Naive Bayes model. The algorithm of the Multinomial Naive Bayes with sub-topics model is shown in Algorithm 1.

In line 2 of Algorithm 1, the user specifies the number of sub-topics. As TAR is a 2-class problem, the number of sub-topics needs to be at least 2. When the number of sub-topics is equal to 2, then the algorithm becomes the semi-supervised Multinomial Naive Bayes model. When the number of sub-topics is 3 or higher, it becomes a semi-supervised Multinomial Naive Bayes with sub-topics. In line 5, the labeled documents are separated from the unlabeled documents. The predefined number of sub-topics are created in line 8. In the case of more than 2 sub-topics, this divides the irrelevant class into subclasses, which is the many-to-one correspondence. In lines 10 to 16, the subclass values are created for the labeled documents. When a document is relevant, only the subclass belonging to the relevant class is assigned a 1. When a document is irrelevant, all subclasses belonging to the irrelevant class get a random probability distribution over the sub-topics, which all sum up to 1. In line 19, the model is fitted based on the labeled documents, including their sub-topics. Following this up, in line 20, the unlabeled documents get their subclass assignment based on the predicted class probability of the fitted model.

Starting from line 23 in Algorithm 1, the unsupervised loop starts, containing the E- and M-steps of Expectation Maximization. The theory of this model is discussed in Section 4.3.4. First, the labeled and unlabeled data are combined in lines 25 and 26. In line 28, the joint log-likelihood score is calculated based on the entire dataset. In line 31, the model is fitted based on the (new) estimated sub-topic assignments (M-Step). In lines 34 to 42, the retrained model estimates the subclass assignments again (E-Step). Here, the labeled documents are handled separately from the unlabeled documents. This is because the class label is already known for the labeled documents. This allows us to set the relevant sub-topic to 1 and all irrelevant sub-topics to 0 if the document is relevant (line 38) or to set the relevant sub-topic to 0 and normalize the irrelevant sub-topic assignments (line 40). Finally, the joint log-likelihood score is calculated again in line 44. Now we can calculate the change in the log-likelihood score for this EM iteration (line 45). After this, the Expectation Maximization loop starts again if the change is larger than ϵ (line 23). When this is the case, the newly predicted subclass assignments (E-Step) are used to retrain the model (M-Step).

When the EM loop has converged, the Multinomial Naive Bayes with sub-topics model is used in the TAR cycle to calculate the class probabilities of the unlabeled documents. This is done through Formula 4.8, where the class probability of a document is calculated by summing all the sub-topic probabilities for the irrelevant class. The relevant class only has one sub-topic, so this sub-topic probability is used for the relevant probability of the document.

SVM with self-training

The ideal situation would be implementing a Semi-Supervised SVM (S3VM) model, especially meanSVM. Unfortunately, as of our knowledge, no implementations based on a research paper of these models are available. Furthermore, the SVM implementation of scikit-learn is based on LIBSVM, which makes it infeasible to adjust this implementation into meanSVM (Pedregosa et al., 2011). Therefore, the wrapper method self-training is implemented to turn the SVM model into a semi-supervised learning model.

Algorithm 1 Expectation Maximization with many-to-one correspondence

```
 1: Initialize Parameters:
 2: number_of_subtopics ← number of sub-topics.                    ▷ specified by user
 3:
 4: Data Preparation:
 5: X_labeled, X_unlabeled, y_labeled, y_unlabeled ← X, y
 6:
 7: Create many-to-one correspondence:
 8: Create a total of number_of_subtopics classes, which represent the sub-topics.
 9:
10: for each document in X_labeled do
11:     if y_labeled = irrelevant then
12:         y_subtopic ← Assign random values to all the irrelevant sub-topics and
    assign 0 to the relevant sub-topic.
13:     else if y_labeled = relevant then
14:         y_subtopic ← Assign 1 to relevant sub-topic and 0 to irrelevant sub-topics.
15:     end if
16: end for
17:
18: Model Initialization:
19: Fit the model on the labeled data (X_labeled, y_subtopic). ▷ Formula 4.4, 4.5 & 4.6
20: y_unlabeled ← Predicted sub-topic probabilities of X_unlabeled.     ▷ Formula 4.7
21:
22: EM (semi-supervised learning) loop:
23: while change > $\epsilon$ do
24:     Combine labeled and unlabeled data:
25:     X ← X_labeled + X_unlabeled
26:     y ← y_subtopic + y_unlabeled
27:
28:     jll_before ← joint log likelihood score of the entire dataset
29:
30:     M-Step:
31:     Fit the model on the combined data (X, y).          ▷ Formula 4.4, 4.5 & 4.6
32:
33:     E-Step:
34:     y_unlabeled ← Predicted sub-topic probabilities of X_unlabeled. ▷ Formula 4.7
35:     y_subtopic ← Predicted sub-topic probabilities of X_labeled.    ▷ Formula 4.7
36:     for each label in y_labeled do              ▷ Correcting labeled probabilities
37:         if label = relevant then
38:             y_subtopic[index of label] ← Set relevant sub-topic to 1 and irrelevant
    sub-topics to 0.
39:         else if label = irrelevant then
40:             y_subtopic[index of label] ← Set relevant sub-topic to 0 and normalise
    irrelevant sub-topics.
41:         end if
42:     end for
43:
44:     jll_after ← joint log likelihood score of the entire dataset
45:     change ← jll_after - jll_before
46: end while
```

The self-training loop adds the top k documents as relevant documents due to the imbalance in the data. When the model is allowed to calculate the class probabilities and add the top k documents to the predicted class, then most, if not all, unlabeled documents would be added to the irrelevant class, as in most cases, less than 1% of the documents are relevant. The implementation of self-training can be seen in Algorithm 2).

In line 2 of Algorithm 2, the user specifies the number of self-training iterations. In line 3, the user specifies the number of unlabeled documents that will be added to the labeled set per iteration. The labeled and unlabeled data are separated in line 6. In line 9, the SVM classifier is fitted on the labeled data. Starting from line 12, the self-training loop starts. In line 13, the decision boundary found by fitting the model on the labeled data is used to calculate the distance between the unlabeled documents and the decision boundary. These values are used in line 14 to find the top k documents with the highest distance to this decision boundary in the direction of the relevant class. Lines 17 to 19 add the top k documents to the labeled documents with a relevant label and remove these top k documents from the unlabeled set. Finally, in line 21, the model is fitted again on the expanded labeled data.

Algorithm 2 SVM with self-training

```
1:  Initialize Parameters:
2:  number_of_iterations ← number of self-training iterations.   ▷ specified by user
3:  k ← number documents added per iteration.                    ▷ specified by user
4:
5:  Data Preparation:
6:  X_labeled, X_unlabeled, y_labeled ← X, y
7:
8:  Model Initialization:
9:  Fit the model on the labeled data (X_labeled, y_labeled).
10:
11: Self-training Loop:
12: for each iteration in number_of_iterations do
13:     decision ← distance of unlabeled documents to the decision boundary
14:     top_k ← top k documents of relevant class with highest decision value
15:
16:     Adding the top k documents to the labeled set as relevant documents
17:     X_labeled ← X_labeled + top_k
18:     y_labeled ← y_labeled + relevant labels for the top_k documents
19:     X_unlabeled ← X_unlabeled without the top_k documents
20:
21:     Fit the model on the expanded labeled data (X_labeled, y_labeled).
22: end for
```

CombiTAR

The final semi-supervised learning model combines AutoTAR, which uses logistic regression, with semi-supervised Multinomial Naive Bayes with sub-topics. The model with Multinomial Naive Bayes with sub-topics is chosen instead of the Multinomial Naive Bayes model because it should perform better with many irrelevant documents, which is the case in the experimental evaluation. This is done by using the supervised AutoTAR classifier at the start of the TAR process and switching to the semi-supervised Multinomial Naive Bayes with sub-topics model when a certain threshold is reached. Through this approach, the semi-supervised learning model can start with more labeled documents, which should help the performance of the semi-supervised learning method.

CombiTAR is implemented with two different thresholds. The first implementation is CombiTAR (50%), which runs AutoTAR until 50% of the relevant documents are found. After this, the Multinomial Naive Bayes with sub-topics model takes over until all relevant documents are found. The second implementation is CombiTAR (90%), which has a threshold of 90%. So, the only difference with the first implementation is that AutoTAR finds 90% of the relevant documents. When 90% of the relevant documents are found, semi-supervised Multinomial Naive Bayes with sub-topics model takes over again until all relevant documents are found

5.4.2 The supervised learning models

In order to be able to answer the research question, the semi-supervised learning models are compared to their corresponding supervised learning models. The following supervised learning models are used in this research: Logistic Regression (used in AutoTAR), Multinomial Naive Bayes, and SVM. Table 5.4 shows an overview of the semi-supervised and supervised models.

Semi-supervised model	**Supervised model**
Labelspreading	Logistic Regression (AutoTAR)
Multinomial Naive Bayes	Multinomial Naive Bayes
Multinomial Naive Bayes with sub-topics	Multinomial Naive Bayes
SVM with self-training	SVM
CombiTAR (50%)	Logistic Regression (AutoTAR)
CombiTAR (90%)	Logistic Regression (AutoTAR)

TABLE 5.4: Overview of the semi-supervised and supervised models

5.5 Hyper-parameter tuning

Performing hyper-parameter tuning during a TAR cycle is challenging, especially as TAR is an iterative process where the active learning component adds new (labeled) data into the train set each cycle. Due to this, splitting the data into a train and test set within the TAR process is impossible. Finding the optimal hyper-parameters by running the whole TAR experiments with different hyper-parameters is not scientifically sound. This can bias the results, overly optimize the model, and cause overfitting. Therefore, we have chosen to search for the optimal hyper-parameter values for the classifiers used in the TAR process. This way, it is possible to split the data into a train and test set through k-fold cross-validation. Through this approach, we aim to preserve the robustness of our models and the validity of our results.

For our experimental evaluation, we will compare the performance of two classifiers with each other in the TAR process. As both classifiers have utilized the same k-fold cross-validation on the same data to find their best hyper-parameters, evaluating their relative performance should still be as fair as possible.

Stratified 5-fold cross-validation in combination with gridsearch is used to find the best hyper-parameter values per model per dataset. The stratified approach is chosen to ensure that each fold has both relevant and irrelevant documents. This is important as there are datasets with less than .2% relevant documents.

Since we are dealing with supervised and semi-supervised models, the train set in each fold contains labeled and unlabeled data, with a .8 and .2 ratio, respectively. The supervised models ignore the unlabeled data in the gridsearch. The folds and the unlabeled documents are fixed and, therefore, the same between all different models.

For the SVM models, an exception is made. The SVM models take longer to train than the other models. Therefore, for the SVM models, a stratified 3-fold cross-validation is used. Also, the best hyper-parameter values for the SVM models are only searched for the first dataset, Appenzeller-Herzog_2020. The hyper-parameter values found for this dataset are used for all datasets. As we do this for both the supervised SVM and the semi-supervised SVM model, we keep the performance comparison as fair as possible.

Gridsearch is used to find the best hyper-parameter values per model per dataset. For the scoring, the logistic loss (neg_log_loss) is used. This is done as our data is highly imbalanced. Due to this, using a scoring metric like accuracy would directly result in high accuracy if the model classifies all documents as irrelevant (see Section 5.3.2). In the TAR scenario, a document can be either relevant or irrelevant. This makes the $y \in \{0, 1\}$. The probability estimate is $p = Pr(y = 1)$. The logistic loss is then calculated as follows. If $y = 1$, then $Logloss(y, p) = -log(p)$. When $y = 0$, then $Logloss(y, p) = log(1 - p)$. This scoring is better suited for an imbalanced dataset, as this scoring does not use the (ratio of) correctly classified documents but compares the difference between the log probability estimate of the model with the true label (Scikit-learn, 2023b).

5.5.1 Logistic Regression

Multiple gridsearches are conducted to identify the optimum parameters. The first gridsearch is performed on the dataset "Appenzeller-Herzog_2020". In this gridsearch, all solvers are used ('lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'), combined with all C values ∈ [.1, .2, .3, ..., .7, .8, .9] and all C values ∈

[1, 2, 3, ..., 98, 99, 100] (Zhu et al., 2011; Fan et al., 2008; Yu, Huang, and Lin, 2011; Schmidt, Le Roux, and Bach, 2017; Defazio, Bach, and Lacoste-Julien, 2014). The results showed that the solver liblinear was the highest-scoring solver. Based on this result and the fact that lbfgs is the default solver, only the liblinear and lbfgs solvers are used in the gridsearch of the following datasets. So, for all remaining datasets, the gridsearch is performed with the solvers 'lbfgs' and 'liblinear' in combination with the same C values as before. The results of the gridsearches are shown in Table 5.5. For more detailed gridsearch results per dataset, see Table A.2 in Appendix A.

5.5.2 Labelspreading

For label spreading, the first gridsearch is performed on all datasets with the kernels 'k-nearest neighbors' (knn) and 'radial basis function' (rbf), and all gamma values between 1 and 50 in the case of kernel rbf and all n_neighbors values between 1 and 50 in the case of knn kernel. On two datasets, the best results were n_neighbor values 49 and 50. Due to this being very close to the edge of the gridsearch, an additional gridsearch was performed on these datasets with kernel knn and n_neighbors values between 50 and 70. See Table 5.5 for an overview of the results.

5.5.3 Multinomial Naive Bayes

The Multinomial Naive Bayes model had no hyper-parameters that needed to be tuned. The model uses Laplace smoothing, which is set by the parameter $\alpha = 1$. Still, the gridsearch was run once per dataset. This way, the results of the logistic loss score can still be compared.

5.5.4 Semi-supervised Multinomial Naive Bayes

The semi-supervised Multinomial Naive Bayes model has no hyper-parameters that need to be tuned. Like the Multinomial Naive Bayes model, the semi-supervised Multinomial Naive Bayes uses Laplace smoothing, and the gridsearch is run once per dataset.

5.5.5 Semi-supervised Multinomial Naive Bayes with sub-topics

As a first step, a gridsearch is done with various number of sub-topics (2, 5, 10, 15, 20, ..., 140, 145, 150). Two datasets showed an optimal number of sub-topics of 150, which is on the edge of our gridsearch. Therefore, an additional exploratory gridsearch was done for these two datasets with varying number of sub-topics (150, 160, 170, ..., 480, 490, 500). Then, a final fine gridsearch was performed on all datasets around the highest-scoring number of sub-topics. The results are shown in Table 5.5. An overview of the best number of sub-topics for each dataset can be found in Table A.1 in Appendix A.

5.5.6 SVM

As mentioned in Section 5.5, the gridsearch is only performed on the first dataset for the SVM models because they take a long time to train. However, by applying the same strategy for both SVM and SVM with self-training, the performance comparison should be as fair as possible, given the time restrictions.

For the SVM model, a gridsearch is done with all C's between $\in [1, 2, 3, ..., 98, 99, 100]$ and gamma equal to 'auto' or 'scale'. When gamma is set to 'auto', then gamma is calculated by $1/\text{number of features}$. When 'scale' is selected, gamma is calculated by $1/(\text{number of features} \times \text{variance of the dataset})$. The results are shown in Table 5.5.

5.5.7 SVM with self-training

The gridsearch of SVM with self-training is also only performed on the first dataset "Appenzeller-Herzog_2020". However, a different gridsearch strategy was applied since more parameters needed to be tuned. In the first gridSearch, various C values have been (1, 10, 20, 30, ..., 80, 90, 100), combined with gamma equal to 'auto' or 'scale' and the number of iterations $\in [1, 2, 3, 4, 5]$.

The best result was C equal to 100, gamma equal to 'auto', and the number of iterations equal to 1. As a C of 100 is on the edge of our gridsearch, a second gridsearch is done with various higher values for C (100, 110, 120, ..., 180, 190, 200), with gamma equal to 'auto' and the number of iterations equal to 1. Again, a C of 100 returned the best results. Therefore, a final, more fine gridsearch is performed with all C values $\in [90, 91, 92, ..., 108, 109, 110]$. The results can be seen in Table 5.5.

Model	**Parameter**	**Mean**	**SD**	**# Datasets**
Logistic Regression *(solver = lbfgs)*	C	9.50	2.29	4
Logistic Regression *(solver = liblinear)*	C	14.25	2.11	8
Label spreading *(kernel = rbf)*	gamma	11.50	1.50	2
Label spreading *(kernel = knn)*	n_neighbors	26.00	13.59	10
MNB	-	-	-	12
SS MNB	-	-	-	12
SS MNB with sub-topics	nr_subtopics	118.83	114.55	12
SVM *(gamma = scale)*	C	3.00	0.00	1
SVM with self-training *(gamma = auto)*	C	100.00	0.00	1
	k_best	1.00	0.00	1
	nr_iterations	1.00	0.00	1

TABLE 5.5: Overview of the gridsearch results with best parameters per model. SS stands for semi-supervised, MNB stands for Multinomial Naive Bayes, and SVM stands for Support Vector Machine.

5.6 Experiments

The research question of this master's thesis is "Can semi-supervised learning be used within Technology Assisted Review to improve the work saved over sampling score?". In order to answer this research question, we will compare the performance of different supervised learning classifiers with their semi-supervised counterparts in TAR. More information about the chosen SSL methods can be found in Section 5.4.

5.6.1 Framework

The framework used for the TAR experiments is the framework of Bron (2021b), which uses the active learning package allib (Bron, 2021a). This is a framework currently being developed by the National Police Lab AI. This police lab is a collaborative initiative of the Dutch Police, Utrecht University, University of Amsterdam, and Delft University of Technology (ICAI, n.d.). The allib package is, as of this moment, not publicly available.

The framework implements different TAR systems, like AutoTAR, ASReview, and TAR systems based on different Machine Learning methods. However, these are all based on supervised learning methods. The different semi-supervised learning methods discussed in section 5.4.1 are implemented into this framework.

5.6.2 Technology Assisted Review

The TAR experiments within the framework of Bron (2021b) work as follows. First, one relevant and one irrelevant document are randomly selected as labeled documents. When a supervised learning method is chosen, the classifier is trained on only these two labeled documents. In the case of a semi-supervised learning method, the classifier is trained on both these two labeled documents and the remaining unlabeled documents. After training the classifier, the classifier predicts the class probability for all unlabeled documents. This returns a probability per document on whether it belongs to the relevant or irrelevant class. The top 100 documents with the highest probability of being relevant are simulated to be shown to a human expert. As we already know all the labels in our experiments, we directly return the correct labels of these 100 documents. After this point, the loop starts again. The now 102 labeled documents are used to train the classifier, after which the classifier predicts the class probability of all unlabeled documents again. This loops until all relevant documents are found. Afterward, the work saved over sampling score is measured when the recall is 95% and when the recall is 100%.

It is important to note that the experimental results depend on the two labeled documents selected at the start. Due to this, the experiments are performed ten times per dataset for all classifiers, except for both SVM classifiers. The experiments are performed six times for the SVM classifiers because they need more training time.

By performing the experiments multiple times and averaging the results, we ensure the reliability of the results. Also, the same ten seeds are used for each classifier. This way, the various classifiers that are compared start with the same two labeled documents in each experiment to ensure a fair comparison (Liu, Timsina, and El-Gayar, 2018).

5.7 Ethics and Privacy

The Ethics and Privacy Quick Scan of the Utrecht University Research Institute of Information and Computing Sciences was conducted (see Appendix C). It classified this research as low-risk with no fuller ethics review or privacy assessment required.

Chapter 6

Results

In this chapter, we present the results of the experimental evaluations to answer the research question, "Can semi-supervised learning be used within Technology Assisted Review to improve the work saved over sampling score?". In order to be able to answer this question, we will have to answer the following sub-questions:

SQ 1. Does the semi-supervised Multinomial Naive Bayes model improve the work saved over sampling score compared to Multinomial Naive Bayes model in Technology Assisted Review?

SQ 2. Does the semi-supervised Multinomial Naive Bayes model with sub-topics improve the work saved over sampling score compared to Multinomial Naive Bayes model in Technology Assisted Review?

SQ 3. Does the SVM with self-training model improve the work saved over sampling score compared to the SVM model in Technology Assisted Review?

SQ 4. Does the label spreading model improve the work saved over sampling score compared to AutoTAR in Technology Assisted Review?

SQ 5. Does combining AutoTAR with semi-supervised Multinomial Naive Bayes with sub-topics improve the work saved over sampling score compared to AutoTAR in Technology Assisted Review?

Each sub-question compares one of the implemented semi-supervised learning models to their corresponding supervised learning variant or the state-of-the-art AutoTAR method.

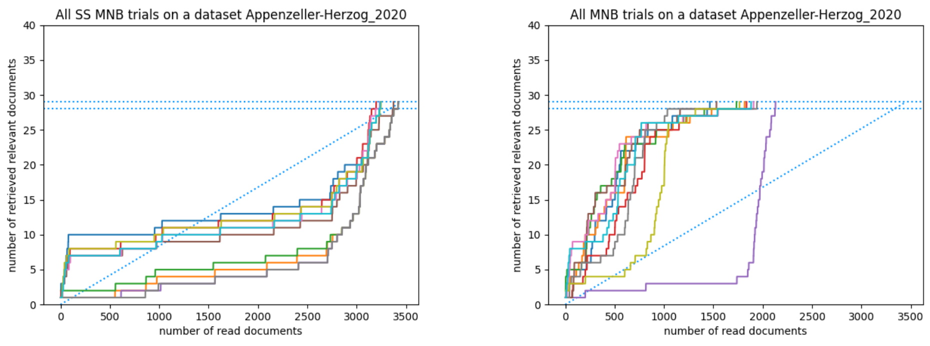
In order to answer the sub-questions, a statistical test is needed to test whether the differences in performance of the two classifiers are different from zero. The experimental evaluations that will be compared across different machine learning models use the same random samples each trial, which allows for a paired test (Demšar, 2006).

The paired t-test should not be used, as the assumption that the differences in scores follow a normal distribution is not guaranteed. Also, the t-test is affected by outliers. Therefore, a non-parametric alternative is used. The Wilcoxon signed-ranks test is a more robust, non-parametric test, as it does not assume normal distributions, and outliers have less effect on the Willcoxon test than the t-test. The Wilcoxon signed-ranks test compares the ranks and the two classifiers' performance differences. The W-Statistic represents the lowest sum of ranks value. First, the differences are ranked according to their absolute values. Then the ranks are summed for the cases where the second model outperformed the first (R^+). The ranks are also summed for the cases where the first model outperforms the second (R^-). Whichever of these two values is lower is reported as the W-statistic value. Through the Wilcoxon signed-ranks test, we attempt to reject the null hypothesis that there is no difference in performance (Demšar, 2006).

6.1 Experiment 1: Semi-supervised Multinomial Naive Bayes and Multinomial Naive Bayes models

For each of the twelve datasets, the semi-supervised Multinomial Naive Bayes and Multinomial Naive Bayes models were used to run ten TAR trials. Each of the ten trials started with the same initial two documents. A Wilcoxon signed-ranks test was performed to identify whether there is a significant difference in performance between the two models. This was done for both the work saved over sampling scores at a recall of 95% (see Table 6.1) and at a recall of 100% (see Table 6.2). The Wilcoxon signed-ranks test revealed a significant difference on all datasets at a recall of 95% (WSS@95%). For the recall of 100% (WSS@100%), the Wilcoxon signed-ranks test revealed a significant difference for all datasets except the Bannach-Brown_2019 (Ban) dataset. On all the datasets where a significant difference was found, the semi-supervised Multinomial Naive Bayes model scored lower in all ten trials than the Multinomial Naive Bayes model (W = 0.0, p = .002).

Figure 6.1 illustrates the performance of the ten trials for the first dataset. The three blue dotted lines each represent a different benchmark. The highest horizontal line represents the threshold for a recall of 100%, the second horizontal line represents the threshold for a recall of 95%, and the diagonal line represents the expected performance of a random sampling strategy. Figure 6.1 shows that the Multinomial Naive Bayes model (6.1(b)) was able to identify more relevant documents earlier in the TAR process and achieved a recall of both 95% and 100% sooner than the semi-supervised model. Detailed graphs for all experiments are available in Appendix B.



(a) Semi-supervised Multinomial Naive Bayes (b) Multinomial Naive Bayes

FIGURE 6.1: All trials for Semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Appenzeller-Herzog_2020.

6.2 Experiment 2: Semi-supervised Multinomial Naive Bayes with sub-topics and Multinomial Naive Bayes models

For each of the twelve datasets, the semi-supervised Multinomial Naive Bayes with sub-topics and the Multinomial Naive Bayes models were used to run ten TAR trials. Again, each of the ten trials started with the same initial two documents. A Wilcoxon signed-ranks test was performed to identify whether there is a significant difference in performance between the two models. This was done for both the work saved over sampling scores at a recall of 95% (see Table 6.3) and at a recall of 100% (see Table 6.4). The Wilcoxon signed-ranks test revealed a significant difference on all datasets for the WSS@95%. It also revealed a significant difference for 11 out of 12 datasets for the WSS@100%.

For the WSS@95%, the semi-supervised Multinomial Naive Bayes with sub-topics model outperformed the Multinomial Naive Bayes model three times. In the remaining nine datasets, its performance was worse. For the WSS@100%, the semi-supervised Multinomial Naive Bayes with sub-topics model outperformed the Multinomial Naive Bayes model five out of eleven times but performed worse on the remaining six datasets.

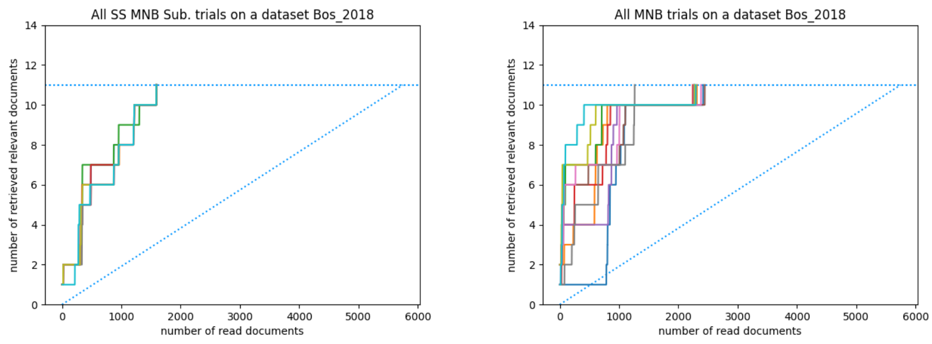
Figure 6.2 illustrates the performance of the ten trials for the dataset Bos_2018. The semi-supervised Multinomial Naive Bayes with sub-topics model (6.2(a)) could find all relevant documents sooner in most cases. There is only one trial where the Multinomial Naive Bayes model (6.2(b)) found all the relevant documents sooner. The graphs also illustrate that the semi-supervised model was able to more consistantly find the relevant documents per trial. There is less variance between the trials.

Datasets	**SS MNB**		**MNB**		**Higher**	**Lower**	**Tie**	***p***	**W**
	M	*SD*	*M*	*SD*					
App.	.013	.026	.543	.084	0	**10**	0	.002	0.0
Ban.	.256	.063	.404	.020	0	**10**	0	.002	0.0
Bos.	.061	.031	.614	.060	0	**10**	0	.002	0.0
Dis.	-.008	.008	.416	.012	0	**10**	0	.002	0.0
Hal.	.276	.088	.884	.004	0	**10**	0	.002	0.0
Kit.	.002	.017	.120	.043	0	**10**	0	.002	0.0
Kwo.	.114	.035	.367	.027	0	**10**	0	.002	0.0
Nag.	.173	.040	.460	.038	0	**10**	0	.002	0.0
Rad.	.080	.076	.756	.022	0	**10**	0	.002	0.0
Scho.	.087	.019	.697	.050	0	**10**	0	.002	0.0
Wah.	.006	.014	.688	.010	0	**10**	0	.002	0.0
Wol.	.007	.003	.405	.053	0	**10**	0	.002	0.0

TABLE 6.1: Comparison of WSS@95% results for semi-supervised Multinomial Naive Bayes and Multinomial Naive Bayes models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@95% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	**SS MNB**		**MNB**		**Higher**	**Lower**	**Tie**	***p***	**W**
	M	*SD*	*M*	*SD*					
App.	.038	.028	.478	.056	0	**10**	0	.002	0.0
Ban.	.133	.006	.143	.022	3	7	0	.193	14.0
Bos.	.061	.031	.614	.060	0	**10**	0	.002	0.0
Dis.	.003	.008	.220	.074	0	**10**	0	.002	0.0
Hal.	.098	.039	.508	.012	0	**10**	0	.002	0.0
Kit.	.026	.010	.115	.019	0	**10**	0	.002	0.0
Kwo.	.081	.018	.141	.032	0	**10**	0	.002	0.0
Nag.	.104	.016	.338	.033	0	**10**	0	.002	0.0
Rad.	.054	.025	.708	.035	0	**10**	0	.002	0.0
Scho.	.064	.034	.716	.043	0	**10**	0	.002	0.0
Wah.	.019	.021	.420	.016	0	**10**	0	.002	0.0
Wol.	.007	.003	.405	.053	0	**10**	0	.002	0.0

TABLE 6.2: Comparison of WSS@100% results for semi-supervised Multinomial Naive Bayes and Multinomial Naive Bayes models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@100% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' and the column 'W' shows the W-statistic value shows the p-value from the Wilcoxon signed-ranks test.



(a) Semi-supervised Multinomial Naive Bayes with sub-topics

(b) Multinomial Naive Bayes

FIGURE 6.2: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Bos_2018.

6.3 Experiment 3: SVM with self-training and SVM models

The datasets van_Dis_2020 (Dis.), Hall_2012 (Hal.), and Wahono_2015 (Wah.) are excluded from the experiments with SVM models. The SVM with self-training and SVM models were used to run six TAR trials for the remaining nine datasets. This was done due to time restrictions. The models started with the same initial two documents for each of these six trials. A Wilcoxon signed-ranks test was performed to identify whether there is a significant difference in performance between the two models. This was done for both the work saved over sampling scores at a recall of 95% (see Table 6.5) and at a recall of 100% (see Table 6.6). The Wilcoxon signed-ranks test found a significant difference for six datasets for WSS@95% and WSS@100%.

For the WSS@95%, the SVM model outperformed the SVM with self-training model six times. For the WSS@100% experiments, the SVM with self-training model outperformed the SVM model one time, while the SVM model outperformed the SVM with self-training model five times.

6.4 Experiment 4: Label spreading and AutoTAR models

The label spreading and AutoTAR models were used to run ten TAR trials for all twelve datasets. Each of the ten trials started with the same initial two documents. A Wilcoxon signed-ranks test was performed to identify whether there is a significant difference in performance between the two models. This was done for both the work saved over sampling scores at a recall of 95% (see Table 6.7) and at a recall of 100% (see Table 6.8). The Wilcoxon signed-ranks test revealed a significant difference in all datasets. For the WSS@95% results, on all the datasets, it can be seen that the label spreading model scored lower in each trial compared to AutoTAR. However, for the WSS@100% results, it can be seen that the label spreading model scores higher

Datasets	SS MNB Sub.		MNB		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.410	.010	.543	.084	1	**9**	0	.004	1.0
Ban.	.341	.007	.404	.020	0	**10**	0	.002	0.0
Bos.	.723	.000	.614	.060	**9**	1	0	.004	1.0
Dis.	.377	.002	.416	.012	0	**10**	0	.002	0.0
Hal.	.470	.025	.884	.004	0	**10**	0	.002	0.0
Kit.	.493	.004	.120	.043	**10**	0	0	.002	0.0
Kwo.	.338	.009	.367	.027	2	**8**	0	.010	3.0
Nag.	.343	.019	.460	.038	0	**10**	0	.002	0.0
Rad.	.506	.006	.756	.022	0	**10**	0	.002	0.0
Scho.	.508	.003	.697	.050	0	**10**	0	.002	0.0
Wah.	.467	.002	.688	.010	0	**10**	0	.002	0.0
Wol.	.532	.000	.405	.053	**10**	0	0	.002	0.0

TABLE 6.3: Comparison of WSS@95% results for semi-supervised Multinomial Naive Bayes with sub-topics and Multinomial Naive Bayes models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@95% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	SS MNB Sub.		MNB		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.204	.084	.478	.056	1	**9**	0	.004	1.0
Ban.	.030	.002	.143	.022	0	**10**	0	.002	0.0
Bos.	.723	.000	.614	.060	**9**	1	0	.004	1.0
Dis.	.303	.000	.220	.074	9	1	0	.084	10.0
Hal.	.449	.001	.508	.012	0	**10**	0	.002	0.0
Kit.	.411	.000	.115	.019	**10**	0	0	.002	0.0
Kwo.	.249	.021	.141	.032	**9**	0	1	.008	0.0
Nag.	.243	.001	.338	.033	0	**10**	0	.002	0.0
Rad.	.350	.000	.708	.035	0	**10**	0	.002	0.0
Scho.	.515	.009	.716	.043	0	**10**	0	.002	0.0
Wah.	.457	.000	.420	.016	**10**	0	0	.002	0.0
Wol.	.532	.000	.405	.053	**10**	0	0	.002	0.0

TABLE 6.4: Comparison of WSS@100% results for semi-supervised Multinomial Naive Bayes with sub-topics and Multinomial Naive Bayes models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@100% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	SS SVM		SVM		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.510	.153	.678	.066	0	**6**	0	.031	0.0
Ban.	.496	.067	.522	.031	3	3	0	.688	8.0
Bos.	.559	.085	.897	.009	0	**6**	0	.031	0.0
Kit.	.487	.059	.532	.009	1	5	0	.156	3.0
Kwo.	.631	.020	.704	.014	0	**6**	0	.031	0.0
Nag.	.618	.023	.600	.083	3	3	0	.562	7.0
Rad.	.747	.022	.784	.029	0	**6**	0	.031	0.0
Scho.	.857	.005	.892	.005	0	**6**	0	.031	0.0
Wol.	.560	.135	.709	.086	0	**6**	0	.031	0.0

TABLE 6.5: Comparison of WSS@95% results for SVM with self-training and SVM models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@95% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	SS SVM		SVM		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.213	.096	.371	.247	2	4	0	.219	4.0
Ban.	.098	.024	.073	.021	5	1	0	.219	4.0
Bos.	.559	.085	.897	.009	0	**6**	0	.031	0.0
Kit.	.050	.030	.129	.033	0	**6**	0	.031	0.0
Kwo.	.255	.025	.499	.094	0	**6**	0	.031	0.0
Nag.	.290	.027	.369	.049	1	5	0	.062	1.0
Rad.	.419	.068	.795	.020	0	**6**	0	.031	0.0
Scho.	.883	.011	.808	.038	**6**	0	0	.031	0.0
Wol.	.560	.135	.709	.086	0	**6**	0	.031	0.0

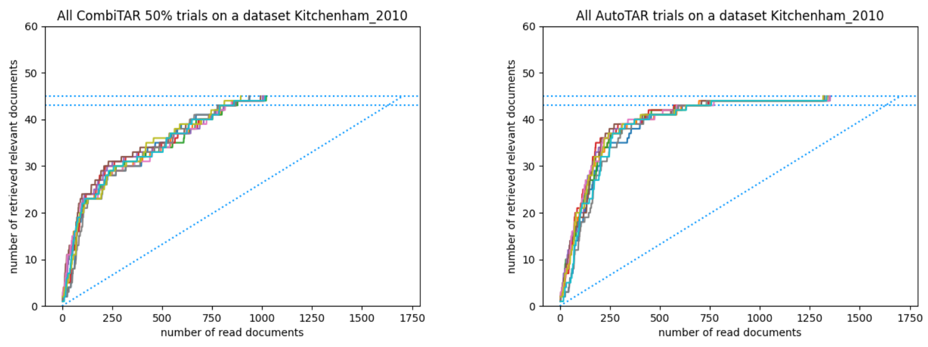
TABLE 6.6: Comparison of WSS@100% results for SVM with self-training and SVM models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@100% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

on each trial for two datasets. The AutoTAR model scored higher on nine datasets on each of the ten trials. Lastly, for the dataset Kitchenham_2010 (Kit.), AutoTAR received a higher WSS@100% score in seven trials, and label spreading received a higher score in three trials. Still, a significant difference was found (W = 6.0, p = .027).

6.5 Experiment 5: CombiTAR and AutoTAR models

The performance of two versions of CombiTAR has been compared to the performance of AutoTAR. The first implementation is CombiTAR 50%. The CombiTAR 50% and AutoTAR models were used to run ten TAR trials for each dataset. All trials started with the same initial two documents. A Wilcoxon signed-ranks test was performed to identify whether there was a significant difference in performance between the two models. This was done for both the work saved over sampling scores at a recall of 95% (see Table 6.9) and at a recall of 100% (see Table 6.10). For the WSS@95% and WSS@100% results, the Wilcoxon signed-ranks test revealed a significant difference in all datasets. For the WSS@95%, the CombiTAR 50% model scored lower in each trial compared to AutoTAR. For the WSS@100% results, the CombiTAR 50% model scored higher for each trial in two datasets and lower for each trial in the remaining nine datasets.

The performance graphs of the Kitchenham_2010 dataset are shown in Figure 6.3. Figure 6.3(a) shows that CombiTAR 50% found fewer relevant documents right after the switch to the semi-supervised model, which happened when 23 relevant documents were found. However, once CombiTAR 50% found 95% of the relevant documents, it required less time to find the remaining relevant documents (100% recall). This illustrates why the CombiTAR 50% model improved the WSS@100% score but not the WSS@95% score.



(a) CombiTAR 50%

(b) AutoTAR

FIGURE 6.3: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Kitchenham_2010.

The second implementation of CombiTAR is CombiTAR 90%. The CombiTAR 90% and AutoTAR models also were used to run ten TAR trials for all datasets. All trials started with the same initial two documents. A Wilcoxon signed-ranks test

Datasets	**Label spr.**		**AutoTAR**		**Higher**	**Lower**	**Tie**	***p***	**W**
	M	*SD*	*M*	*SD*					
App.	.268	.190	.631	.007	0	**10**	0	.002	0.0
Ban.	.368	.026	.497	.014	0	**10**	0	.002	0.0
Bos.	.692	.243	.921	.010	0	**10**	0	.002	0.0
Dis.	.315	.034	.711	.006	0	**10**	0	.002	0.0
Hal.	.843	.011	.921	.001	0	**10**	0	.002	0.0
Kit.	.323	.027	.604	.013	0	**10**	0	.002	0.0
Kwo.	.600	.019	.727	.017	0	**10**	0	.002	0.0
Nag.	.536	.010	.620	.005	0	**10**	0	.002	0.0
Rad.	.764	.016	.825	.005	0	**10**	0	.002	0.0
Scho.	.768	.020	.914	.003	0	**10**	0	.002	0.0
Wah.	.745	.021	.819	.001	0	**10**	0	.002	0.0
Wol.	.307	.221	.769	.032	0	**10**	0	.002	0.0

TABLE 6.7: Comparison of WSS@95% results for label spreading and AutoTAR models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@95% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	**Label spr.**		**AutoTAR**		**Higher**	**Lower**	**Tie**	***p***	**W**
	M	*SD*	*M*	*SD*					
App.	.021	.056	.325	.166	0	**10**	0	.002	0.0
Ban.	.021	.012	.056	.007	0	**10**	0	.002	0.0
Bos.	.692	.243	.921	.010	0	**10**	0	.002	0.0
Dis.	.234	.070	.620	.002	0	**10**	0	.002	0.0
Hal.	.588	.010	.565	.000	**10**	0	0	.002	0.0
Kit.	.110	.112	.218	.008	3	**7**	0	.027	6.0
Kwo.	.004	.001	.296	.005	0	**10**	0	.002	0.0
Nag.	.005	.002	.438	.008	0	**10**	0	.002	0.0
Rad.	.001	.000	.843	.000	0	**10**	0	.002	0.0
Scho.	.785	.016	.857	.003	0	**10**	0	.002	0.0
Wah.	.540	.045	.419	.001	**10**	0	0	.002	0.0
Wol.	.307	.221	.769	.032	0	**10**	0	.002	0.0

TABLE 6.8: Comparison of WSS@100% results for label spreading and AutoTAR models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@100% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

was performed to identify whether there is a significant difference in performance between the two models. This was done for both the work saved over sampling scores at a recall of 95% (see Table 6.9) and at a recall of 100% (see Table 6.10). For the WSS@95% results, the Wilcoxon signed-ranks test revealed a significant difference for ten datasets. The CombiTAR 90% model scored lower in each trial than AutoTAR for the ten datasets with a significant difference. For the WSS@100% results, the Wilcoxon signed-ranks test revealed a significant difference for all twelve datasets. The CombiTAR 90% model scored higher for each trial in two datasets and lower for each trial in ten datasets.

Datasets	CombiTAR 50%		AutoTAR		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.442	.051	.631	.007	0	**10**	0	.002	0.0
Ban.	.342	.012	.497	.014	0	**10**	0	.002	0.0
Bos.	.730	.014	.921	.010	0	**10**	0	.002	0.0
Dis.	.378	.007	.711	.006	0	**10**	0	.002	0.0
Hal.	.459	.012	.921	.001	0	**10**	0	.002	0.0
Kit.	.490	.008	.604	.013	0	**10**	0	.002	0.0
Kwo.	.334	.019	.727	.017	0	**10**	0	.002	0.0
Nag.	.340	.014	.620	.005	0	**10**	0	.002	0.0
Rad.	.503	.005	.825	.005	0	**10**	0	.002	0.0
Scho.	.543	.027	.914	.003	0	**10**	0	.002	0.0
Wah.	.465	.010	.819	.001	0	**10**	0	.002	0.0
Wol.	.529	.003	.769	.032	0	**10**	0	.002	0.0

TABLE 6.9: Comparison of WSS@95% results for CombiTAR 50% and AutoTAR models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@95% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	CombiTAR 50%		AutoTAR		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.277	.130	.325	.166	0	**10**	0	.002	0.0
Ban.	.028	.002	.056	.007	0	**10**	0	.002	0.0
Bos.	.730	.014	.921	.010	0	**10**	0	.002	0.0
Dis.	.304	.002	.620	.002	0	**10**	0	.002	0.0
Hal.	.451	.004	.565	.000	0	**10**	0	.002	0.0
Kit.	.423	.026	.218	.008	**10**	0	0	.002	0.0
Kwo.	.249	.027	.296	.005	0	**10**	0	.002	0.0
Nag.	.235	.031	.438	.008	0	**10**	0	.002	0.0
Rad.	.361	.034	.843	.000	0	**10**	0	.002	0.0
Scho.	.514	.013	.857	.003	0	**10**	0	.002	0.0
Wah.	.461	.001	.419	.001	**10**	0	0	.002	0.0
Wol.	.529	.003	.769	.032	0	**10**	0	.002	0.0

TABLE 6.10: Comparison of WSS@100% results for CombiTAR 50% and AutoTAR models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@100% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	CombiTAR 90%		AutoTAR		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.443	.051	.631	.007	0	**10**	0	.002	0.0
Ban.	.327	.003	.497	.014	0	**10**	0	.002	0.0
Bos.	.866	.029	.921	.010	0	**10**	0	.002	0.0
Dis.	.426	.001	.711	.006	0	**10**	0	.002	0.0
Hal.	.485	.053	.921	.001	0	**10**	0	.002	0.0
Kit.	.582	.048	.604	.013	5	5	0	.432	19.0
Kwo.	.583	.028	.727	.017	0	**10**	0	.002	0.0
Nag.	.352	.021	.620	.005	0	**10**	0	.002	0.0
Rad.	.495	.004	.825	.005	0	**10**	0	.002	0.0
Scho.	.722	.063	.914	.003	0	**10**	0	.002	0.0
Wah.	.742	.070	.819	.001	0	**10**	0	.002	0.0
Wol.	.527	.003	.769	.032	0	**10**	0	.002	0.0

TABLE 6.11: Comparison of WSS@95% results for CombiTAR 90% and AutoTAR models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@95% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Datasets	CombiTAR 90%		AutoTAR		Higher	Lower	Tie	*p*	W
	M	*SD*	*M*	*SD*					
App.	.275	.128	.325	.166	0	**10**	0	.002	0.0
Ban.	.026	.002	.056	.007	0	**10**	0	.002	0.0
Bos.	.866	.029	.921	.010	0	**10**	0	.002	0.0
Dis.	.311	.001	.620	.002	0	**10**	0	.002	0.0
Hal.	.457	.003	.565	.000	0	**10**	0	.002	0.0
Kit.	.404	.013	.218	.008	**10**	0	0	.002	0.0
Kwo.	.240	.019	.296	.005	0	**10**	0	.002	0.0
Nag.	.230	.030	.438	.008	0	**10**	0	.002	0.0
Rad.	.354	.002	.843	.000	0	**10**	0	.002	0.0
Scho.	.538	.045	.857	.003	0	**10**	0	.002	0.0
Wah.	.492	.004	.419	.001	**10**	0	0	.002	0.0
Wol.	.527	.003	.769	.032	0	**10**	0	.002	0.0

TABLE 6.12: Comparison of WSS@100% results for CombiTAR 90% and AutoTAR models. The Mean (*M*) and Standard Deviation (*SD*) of the WSS@100% values are shown. The 'Higher', 'Lower', and 'Tie' represent the number of times where the semi-supervised model scored higher, lower, or equal to the supervised model, respectively. Column 'p' shows the p-value and the column 'W' shows the W-statistic value from the Wilcoxon signed-ranks test.

Chapter 7

Discussion

This study aims to investigate whether semi-supervised learning can be used within Technology Assisted Review to further minimizing the reviewers' effort while maximizing the recall. The evaluation metric work saved over sampling is used to measure this. For this investigation, several semi-supervised and supervised learning methods are implemented into the Technology Assisted Review framework of the National Police LAB AI (Bron, 2021b). A total of five different semi-supervised learning models are compared to their corresponding supervised learning models across twelve different datasets.

7.1 Research questions

The research question of this study is: "Can semi-supervised learning be used within Technology Assisted Review to improve the work saved over sampling score?". The sub-questions of this study each focus on a different semi-supervised learning method and compare the performance of this model to their supervised counterpart.

The first sub-question is "SQ 1. Does the semi-supervised Multinomial Naive Bayes model improve the work saved over sampling score compared to the Multinomial Naive Bayes model in Technology Assisted Review?". For this sub-question, the semi-supervised Multinomial Naive Bayes model, which utilizes Expectation Maximization as described by Nigam, McCallum, and Mitchell (2006), is compared to the supervised version of Multinomial Naive Bayes.

The second sub-question is "SQ 2. Does the semi-supervised Multinomial Naive Bayes model with sub-topics improve the work saved over sampling score compared to the Multinomial Naive Bayes model in Technology Assisted Review?". In addition to the semi-supervised Multinomial Naive Bayes model from sub-question one, this model also implements a many-to-one correspondence, as described by Nigam, McCallum, and Mitchell (2006).

The third sub-question is "SQ 3. Does the SVM with self-training model improve the work saved over sampling score compared to the SVM model in Technology Assisted Review?". For this sub-question, the semi-supervised SVM model, which uses the wrapper method self-training, is compared to the supervised version of SVM.

The fourth sub-question is "SQ 4. Does the label spreading model improve the work saved over sampling score compared to AutoTAR in Technology Assisted Review?". For this sub-question, the semi-supervised model label spreading is compared to the AutoTAR model, which uses a logistic regression model. Because label spreading has no supervised counterpart, AutoTAR is chosen to compare the performance against, as this is the current state-of-the-art TAR implementation.

The fifth and final sub-question is "SQ 5. Does combining AutoTAR with semi-supervised Multinomial Naive Bayes with sub-topics improve the work saved over sampling score compared to AutoTAR in Technology Assisted Review?". In order to answer this sub-question, two versions of combining AutoTAR with semi-supervised Multinomial Naive Bayes with sub-topics are compared to AutoTAR. Through this, we aim to find out if combining the supervised learning model with a semi-supervised model can improve the performance of the supervised learning model. In the first version, AutoTAR first finds 50% of the relevant documents before switching to semi-supervised model (CombiTAR 50%). In the second version, the switch towards the semi-supervised model is done when AutoTAR has found 90% of the relevant documents (CombiTAR 90%).

7.2 Findings and interpretation

7.2.1 Sub-questions

For the majority of the TAR experiments, a significant difference was found between the semi-supervised and supervised models based on the work saved over sampling scores.

For the first sub-question, 23 out of 24 experiments resulted in a significant difference. However, for all 23 experiments, the supervised learning model outperformed the semi-supervised Multinomial Naive Bayes model. This shows that the semi-supervised Multinomial Naive Bayes model could not improve the work saved over sampling score compared to the Multinomial Naive Bayes model in Technology Assisted Review. The lower performance of the semi-supervised Multinomial Naive Bayes model could be caused by the model probability not being correlated with the classification accuracy. As discussed in Section 4.3.3, the EM algorithm optimizes on posterior model probability. Therefore, the introduction of unlabeled data may lower classification accuracy if the model probability does not positively correlate to the model's accuracy. In this case, adopting a more expressive generative model can restore this correlation, thereby enhancing the accuracy (Nigam, McCallum, and Mitchell, 2006). The model discussed in sub-question two is a more expressive generative model due to the many-to-one correspondence in the form of sub-topics.

For the second sub-question, 23 out of 24 experiments resulted in a significant difference in performance. For the work saved over sampling at 95% recall

(WSS@95%), the semi-supervised Multinomial Naive Bayes with sub-topics model performed significantly better three out of twelve times. Furthermore, for the work saved over sampling at 100% recall (WSS@100%), the semi-supervised Multinomial Naive Bayes with sub-topics model performed significantly better five out of eleven times. The increase in performance compared to the semi-supervised Multinomial Naive Bayes model from sub-question one could be explained by the increased representativeness of a model due to the addition of sub-topics, which replaced the one-to-one assumption with a less restrictive one (Nigam, McCallum, and Mitchell, 2006). The semi-supervised model performs better in 8 out of 23 experiments. So, it can be concluded that in some situations, the semi-supervised Multinomial Naive Bayes with sub-topics model can improve the work saved over sampling score compared to the supervised Multinomial Naive Bayes model. However, finding the optimal number of sub-topics could be a challenge when the data is not labeled. Interestingly, the semi-supervised model outperforms the supervised model on the two datasets with the lowest inclusion rate, namely 0.19% (Bos_2018) and 0.38% (Wolters_2018) for both the WSS@95% and WSS@100% scores. This could indicate that this semi-supervised model, which uses sub-topics, is able to outperform the supervised model on datasets with a relatively large number of irrelevant documents.

The third sub-question compares the SVM with self-training to the SVM model. Eleven out of eighteen experiments showed a significant difference based on the work saved over sampling scores. In the experiments where a significant difference was found, the supervised SVM model outperformed the SVM with self-training model ten out of eleven times. So, it can be concluded that the SVM with self-training model rarely improves the work saved over sampling score compared to the SVM model. For these results, it is interesting to note that in the hyper-parameter value tuning, the best number of self-training iterations was achieved with only one self-training iteration. This shows that even though only a single self-training iteration is performed for each active learning cycle in the TAR process, the self-training implementation still significantly decreases the WSS score. A possible explanation for this could be that the document with the highest probability of being relevant is added to the labeled dataset as relevant, regardless of the estimated probability. This is done due to the highly skewed data, as discussed in Section 5.4.1.

The fourth sub-question compares the performance of the semi-supervised label spreading model with the AutoTAR model. All 24 experiments resulted in a significant difference in performance. The label spreading model was only able to perform better in 2 out of the 24 experiments. So, it can be concluded that there are datasets in which the label spreading model can improve the work saved over sampling score compared to AutoTAR. However, we could only find this result in two experiments and only for the WSS@100% scores. So, it is not likely to occur. However, it is important to note that label spreading is not compared to its corresponding supervised learning model, as a supervised version of label spreading does not exist. Due to this, the label spreading model is compared to the state-of-the-art AutoTAR model.

The fifth and final sub-question compares the performance of both CombiTAR 50% and CombiTAR 90% to AutoTAR in order to find out whether combining AutoTAR with semi-supervised Multinomial Naive Bayes with sub-topics can improve the work saved over sampling score compared to AutoTAR. For CombiTAR 50%, all 24 experiments resulted in a significant difference. However, CombiTAR 50% could only outperform AutoTAR on two datasets for only the WSS@100% scores. The results of CombiTAR 90% showed a significant difference in 23 out of 24 experiments. CombiTAR 90% was also only able to outperform AutoTAR on the same two datasets as CombiTAR 50%, and again only for the WSS@100% scores. So, it can be concluded that it is possible that combining AutoTAR with semi-supervised Multinomial Naive Bayes with sub-topics improves the work saved over sampling score compared to AutoTAR in Technology Assisted Review. However, it is not likely to occur, as this only happened 4 times out of 47.

7.2.2 Research question

The research question of this study was "Can semi-supervised learning be used within Technology Assisted Review to improve the work saved over sampling score?". Sub-questions 1 and 3, involving the semi-supervised Multinomial Naive Bayes model, and the SVM with self-training model showed no results supporting that semi-supervised learning could improve the work saved over sampling score. Sub-question 4, which involved the label spreading model, could only improve the WSS score in 2 out of 24 cases. However, sub-question 2, involving the semi-supervised Multinomial Naive Bayes with sub-topics model, showed more frequent results where the semi-supervised learning model produced higher work saved over sampling scores. Especially interesting is the fact that this model performed better on the two datasets with the lowest rate of relevant documents for both the WSS@95% and WSS@100% scores. This demonstrates that it is possible for the semi-supervised learning models to improve the work saved over sampling score. In the results of sub-question 5, when comparing multiple combinations of AutoTAR with semi-supervised Multinomial Naive Bayes with sub-topics to the supervised AutoTAR model, there are datasets in which the addition of the semi-supervised learning method significantly increased the work saved over sampling score. So, even though the semi-supervised models were not able to outperform the supervised version in the majority of cases, we have shown that it is possible to outperform the supervised learning methods by using semi-supervised models.

7.3 Limitations

A limitation of this research is that it was not possible to tune the hyper-parameter values of the classifiers within the TAR process itself without running the whole TAR process multiple times with different hyper-parameters, as discussed in Section 5.5. Due to this, the hyper-parameter values are tuned through stratified k-fold cross-validation on the fully labeled datasets to preserve the robustness of our models and the validity of our results. This is a limitation as it negatively affects the generalisability of the research results. When TAR is used in a real-life problem, then the labels of the documents still need to be discovered. Therefore, it is impossible to tune the model's hyper-parameter values in the same way as done in this research without labeling all the documents first. However, this approach allowed us to compare the results of the models in a way where both models had the hyper-parameter values found in the best case scenario, which kept the comparison between models as fair as possible. For the model comparison in sub-questions 2 and 4, it is important to note that the models label spreading and Multinomial Naive Bayes with sub-topics do not have a direct supervised version, which makes the comparison more complicated. The supervised Multinomial Naive Bayes model was chosen for the comparison because this is the closest counterpart to the Multinomial Naive Bayes with sub-topics model. Label spreading did not have such a close counterpart. Therefore the state-of-the-art AutoTAR model was chosen for comparison.

Determining the number of relevant documents found for the CombiTAR 50% and CombiTAR 90% models encountered a similar issue to finding the hyper-parameter values. Knowing how many relevant documents were included in the dataset allowed us to see whether improving the WSS scores compared to AutoTAR was possible by setting the threshold for the switch at exactly 50% and 90%. However, in a real-life scenario, it is not known how many relevant documents there are in a dataset. This means that the number of relevant documents must be estimated to calculate the threshold values of 50% or 90% relevant documents. However, the CombiTAR 50% model outperformed the AutoTAR model on the same two datasets as the CombiTAR 90% model. This could indicate that the percentage of relevant documents can be somewhat estimated. Li and Kanoulas (2020) already described several methods to estimate the total number of relevant documents (see Section 3.1).

Another limitation of the generalizability is that in real-life cases, human reviewers make classification errors in the active learning step of the TAR process (Yu and Menzies, 2019). This human error rate is not considered in our experiments. In our experiments, the correct labels were always provided to the classifier. This is a limitation as it is unknown which semi-supervised model is affected more by different percentages of mislabeled documents. However, it is important to note that multiple solutions are already available for reducing human errors in the active learning cycle, like a majority vote or rechecking a few documents (Yu and Menzies, 2019).

The primary limitation of the semi-supervised SVM model is that no implementations based on a research paper were available of meanSVM or semi-supervised SVM at all. Due to time restrictions, it was also not possible to implement these models ourselves. This prohibited us from comparing the performance of meanSVM and SVM. This is a limitation as the literature indicates that meanSVM is the best performing semi-supervised version of SVM for large datasets (Ding, Zhu, and Zhang, 2017). It is possible that meanSVM would have performed significantly better than SVM on the WSS scores.

The final limitation of the SVM models is that training them takes a long time, especially on larger datasets. Due to this, we could not run the TAR trials for the three largest datasets. It was also not possible to run the trials ten times per dataset. This is a limitation because it would be interesting to see if performance differs on a larger dataset. However, running the SVM models on the three largest datasets would have taken multiple weeks of non-stop running experiments. As discussed before, running the SVM models for multiple weeks was not possible due to time restrictions.

7.4 Future research

A few interesting ideas for future research have emerged from this study. The first idea for future research is related to the success of semi-supervised Multinomial Naive Bayes with sub-topics on datasets with the lowest rate of relevant documents. It could be further explored if this model outperforms supervised learning models on other datasets with similarly low rates of relevant documents.

The second idea for future research regarding the Multinomial Naive Bayes with sub-topics model is the sensitivity of the number of sub-topics on the performance. This study found the optimal number of sub-topics based on k-fold cross-validation on the labeled dataset. As this is impossible on unlabeled datasets, it is interesting to know how much a deviating number of sub-topics influences the performance.

Future research could also examine the effects on performance when the relevant class is allowed to have multiple sub-topics. Due to the highly skewed datasets, only a single sub-topic was assigned to the relevant class in this study.

Another idea for future research is to look for a way to tune the hyper-parameter values of a classifier used in the TAR process. The best hyper-parameter values found in this study vary substantially between datasets, as seen in Section 5.5, Table 5.5, and Appendix A. This indicates that it is beneficial to find a way to tune the hyper-parameter values in the TAR process, even if not all the document labels are available. For the hyper-parameter "number of sub-topics," the search queries on which the dataset is based or a domain expert could be utilized to estimate the number of sub-topics.

Future research could also expand the current research by exploring the performance differences between semi-supervised and supervised models on even larger datasets. The largest dataset used in this study contains 10,953 documents.

The five datasets used by Li and Kanoulas (2020) vary between 82,421 and 685,592 documents. These datasets are publicly available at `https://github.com/dli1/auto-stop-tar` (Kanoulas et al., 2017; Kanoulas et al., 2018; Kanoulas et al., 2019).

Finally, future research could expand the current study by exploring different strategies to improve the performance of the models. This can be explored by implementing balance strategies to rebalance the training data. Undersampling can be used to exclude a part of the irrelevant class to balance the classes. Dynamic resampling can be used to increase the relevant class size by duplicating documents (Schoot et al., 2021). It would be interesting to explore how this affects both semi-supervised and supervised learning models.

7.5 Conclusion

This study compared the performance of five semi-supervised learning classifiers against their supervised equivalents to investigate whether semi-supervised learning can be used to improve the work saved over sampling score within Technology Assisted Review. The results showed that semi-supervised learning models are able to improve performance. The semi-supervised Multinomial Naive Bayes classifier, with many-to-one correspondence via sub-topics, showed the most promising results. This model outperformed the supervised model on the two datasets with the lowest inclusion rate. Future work could expand the current research by exploring whether these findings also occur on other datasets with a similarly large number of irrelevant documents and by further investigating ways to improve the performance of the semi-supervised models.

References

ASReview (2022). *Systematic Review Datasets*. URL: https://github.com/asreview/systematic-review-datasets (visited on 11/20/2022).

Bron, Michiel (2021a). *Python package instancelib*. URL: https://github.com/mpbron/instancelib.

— (2021b). *Python package python-allib*. URL: https://github.com/mpbron/allib.

Chawla, N. V. et al. (June 2002). "SMOTE: Synthetic Minority Over-sampling Technique". en. In: *Journal of Artificial Intelligence Research* 16, pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: https://www.jair.org/index.php/jair/article/view/10302 (visited on 10/27/2022).

Cohen, A. M. et al. (Mar. 2006). "Reducing Workload in Systematic Review Preparation Using Automated Citation Classification". In: *Journal of the American Medical Informatics Association* 13.2, pp. 206–219. ISSN: 1067-5027. DOI: 10.1197/jamia.M1929. URL: https://doi.org/10.1197/jamia.M1929 (visited on 10/25/2022).

Cormack, Gordon V. and Maura R. Grossman (Aug. 2015). "Multi-Faceted Recall of Continuous Active Learning for Technology-Assisted Review". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. New York, NY, USA: Association for Computing Machinery, pp. 763–766. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767771. URL: https://doi.org/10.1145/2766462.2767771 (visited on 09/08/2022).

Defazio, Aaron, Francis Bach, and Simon Lacoste-Julien (Dec. 2014). *SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives*. arXiv:1407.0202 [cs, math, stat]. URL: http://arxiv.org/abs/1407.0202 (visited on 06/05/2023).

Demšar, Janez (2006). "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine learning research* 7. Publisher: JMLR. org, pp. 1–30.

Ding, Shifei, Zhibin Zhu, and Xiekai Zhang (2017). "An overview on semi-supervised support vector machine". In: *Neural Computing and Applications* 28.5. Publisher: Springer, pp. 969–978.

Fan, Rong-En et al. (2008). "LIBLINEAR: A Library for Large Linear Classification". en. In: *The Journal of Machine learning research*.

Fernandez, Alberto et al. (Apr. 2018). "SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary". en. In: *Journal of Artificial Intelligence Research* 61, pp. 863–905. ISSN: 1076-9757. DOI: 10.1613/jair

.1 .11192. URL: http://jair.org/index.php/jair/article/view/11192 (visited on 10/27/2022).

ICAI (n.d.). *Police Lab AI*. en-US. URL: https://icai.ai/police-lab-ai/ (visited on 11/29/2022).

Kanoulas, Evangelos et al. (2017). "CLEF 2017 Technologically Assisted Reviews in Empirical Medicine Overview". en. In: vol. 1866. 1–29. CEUR Workshop Proceedings. URL: https://www.researchgate.net/publication/319130742_CLEF_2017_Technologically_Assisted_Reviews_in_Empirical_Medicine_Overview (visited on 11/29/2022).

— (2018). "CLEF 2018 Technologically Assisted Reviews in Empirical Medicine Overview". en. In: CEUR Workshop Proceedings. URL: https://ceur-ws.org/Vol-2125/invited_paper_6.pdf (visited on 11/29/2022).

— (2019). "CLEF 2019 Technology Assisted Reviews in Empirical Medicine Overview". en. In: vol. 2380. CEUR Workshop Proceedings. URL: https://ceur-ws.org/Vol-2380/paper_250.pdf (visited on 11/29/2022).

Li, Dan and Evangelos Kanoulas (Sept. 2020). "When to Stop Reviewing in Technology-Assisted Reviews: Sampling from an Adaptive Distribution to Estimate Residual Relevant Documents". In: *ACM Transactions on Information Systems* 38.4, 41:1–41:36. ISSN: 1046-8188. DOI: 10.1145/3411755. URL: https://doi.org/10.1145/3411755 (visited on 11/06/2022).

Liu, Jun, Prem Timsina, and Omar El-Gayar (2018). "A comparative analysis of semi-supervised learning: the case of article selection for medical systematic reviews". In: *Information Systems Frontiers* 20.2. Publisher: Springer, pp. 195–207.

Nigam, Kamal, Andrew McCallum, and Tom M. Mitchell (Sept. 2006). "Semi-Supervised Text Classification Using EM". en. In: *Semi-Supervised Learning*. Ed. by Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. The MIT Press, pp. 32–55. ISBN: 978-0-262-03358-9. DOI: 10.7551/mitpress/9780262033589.003.0003. URL: https://academic.oup.com/mit-press-scholarship-online/book/41571/chapter/353090558 (visited on 09/01/2022).

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Schmidt, Mark, Nicolas Le Roux, and Francis Bach (Mar. 2017). "Minimizing finite sums with the stochastic average gradient". en. In: *Mathematical Programming* 162.1-2, pp. 83–112. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-016-1030-6. URL: http://link.springer.com/10.1007/s10107-016-1030-6 (visited on 06/05/2023).

Schoot, Rens van de et al. (Feb. 2021). "An open source machine learning framework for efficient and transparent systematic reviews". en. In: *Nature Machine Intelligence* 3.2, pp. 125–133. ISSN: 2522-5839. DOI: 10.1038/s42256-020-00287-7. URL: http://www.nature.com/articles/s42256-020-00287-7 (visited on 09/30/2022).

Scikit-learn (2023a). *1.14. Semi-supervised learning*. en. URL: https://scikit-learn/stable/modules/semi_supervised.html (visited on 05/31/2023).

— (2023b). *sklearn.metrics.log_loss*. en. URL: https://scikit-learn/stable/modules/generated/sklearn.metrics.log_loss.html (visited on 05/31/2023).

Søgaard, Anders (May 2013). *Semi-Supervised Learning and Domain Adaptation in Natural Language Processing*. en. Google-Books-ID: nexcAQAAQBAJ. Morgan & Claypool Publishers. ISBN: 978-1-60845-986-5.

Yang, Eugene, David D. Lewis, and Ophir Frieder (Aug. 2021). "On minimizing cost in legal document review workflows". In: *Proceedings of the 21st ACM Symposium on Document Engineering*. DocEng '21. New York, NY, USA: Association for Computing Machinery, pp. 1–10. ISBN: 978-1-4503-8596-1. DOI: 10.1145/3469096.3469872. URL: https://doi.org/10.1145/3469096.3469872 (visited on 11/21/2022).

Yu, Hsiang-Fu, Fang-Lan Huang, and Chih-Jen Lin (Oct. 2011). "Dual coordinate descent methods for logistic regression and maximum entropy models". en. In: *Machine Learning* 85.1-2, pp. 41–75. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-010-5221-8. URL: http://link.springer.com/10.1007/s10994-010-5221-8 (visited on 06/05/2023).

Yu, Zhe and Tim Menzies (Apr. 2019). "FAST2: An intelligent assistant for finding relevant papers". en. In: *Expert Systems with Applications* 120, pp. 57–71. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.11.021. URL: https://www.sciencedirect.com/science/article/pii/S0957417418307413 (visited on 09/01/2022).

Zhu, Ciyou et al. (2011). *L-BFGS-B: Software for Large-scale Bound-constrained Optimization*. URL: https://users.iems.northwestern.edu/~nocedal/lbfgsb.html (visited on 06/05/2023).

Zhu, Xiaojin and Andrew B. Goldberg (2009). *Introduction to Semi-Supervised Learning*. en. Cham: Springer International Publishing. ISBN: 978-3-031-00420-9 978-3-031-01548-9. DOI: 10.1007/978-3-031-01548-9. URL: https://link.springer.com/10.1007/978-3-031-01548-9 (visited on 09/27/2022).

Appendix A

Hyper-parameter values

Name	Number of sub-topics
Appenzeller-Herzog_2020	94
Bannach-Brown_2019	6
Bos_2018	434
Hall_2012	85
Kitchenham_2010	35
Kwok_2020	20
Nagtegaal_2019	20
Radjenovic_2013	113
van_Dis_2020	133
van_de_Schoot_2017	134
Wahono_2015	105
Wolters_2018	247

TABLE A.1: Overview of the best found hyper-parameter values per dataset for the semi-supervised Multinomial Naive Bayes with sub-topics model.

Name	Solver	C
Appenzeller-Herzog_2020	liblinear	16
Bannach-Brown_2019	liblinear	16
Bos_2018	lbfgs	13
Hall_2012	liblinear	14
Kitchenham_2010	liblinear	11
Kwok_2020	liblinear	15
Nagtegaal_2019	liblinear	11
Radjenovic_2013	liblinear	14
van_Dis_2020	lbfgs	7
van_de_Schoot_2017	liblinear	17
Wahono_2015	lbfgs	8
Wolters_2018	lbfgs	10

TABLE A.2: Overview of the best found hyper-parameter values for each dataset for the Logistic Regression model.

Name	**Kernel**	**n_neighbors**	**Gamma**
Appenzeller-Herzog_2020	K-Nearest Neighbors	18	-
Bannach-Brown_2019	K-Nearest Neighbors	10	-
Bos_2018	Radial Basis Functions	-	13
Hall_2012	K-Nearest Neighbors	14	-
Kitchenham_2010	K-Nearest Neighbors	31	-
Kwok_2020	K-Nearest Neighbors	23	-
Nagtegaal_2019	K-Nearest Neighbors	25	-
Radjenovic_2013	K-Nearest Neighbors	24	-
van_Dis_2020	K-Nearest Neighbors	51	-
van_de_Schoot_2017	K-Nearest Neighbors	50	-
Wahono_2015	K-Nearest Neighbors	14	-
Wolters_2018	Radial Basis Function	-	10

TABLE A.3: Overview of the best found hyper-parameter values for each dataset for the label spreading model.

Name	**Gamma**	**C**
Appenzeller-Herzog_2020	scale	3

TABLE A.4: Overview of the best found hyper-parameter values for the Support Vector Machine model, which uses the Radial Basis Function (RBF) kernel.

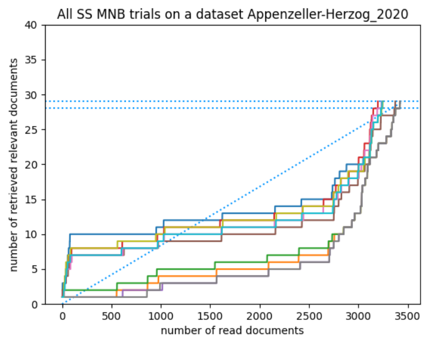
Name	**Gamma**	**C**	**Nr_iterations**	**K_best**
Appenzeller-Herzog_2020	auto	100	1	1

TABLE A.5: Overview of the best found hyper-parameter values for the model Support Vector Machine with Self-Training, which uses the Radial Basis Function (RBF) kernel.

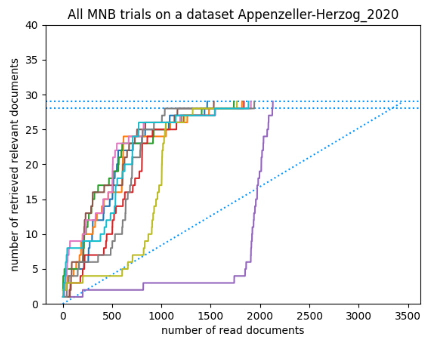
Appendix B

Graphs of all the TAR trials

B.1 Experiment 1: Semi-supervised Multinomial Naive Bayes and Multinomial Naive Bayes models

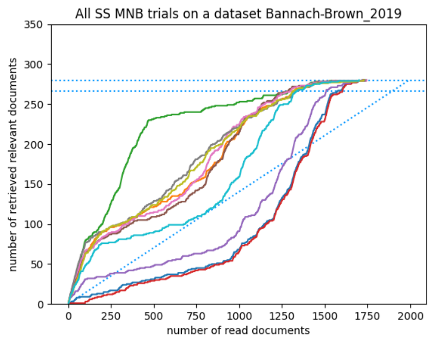


(a) Semi-supervised Multinomial Naive Bayes

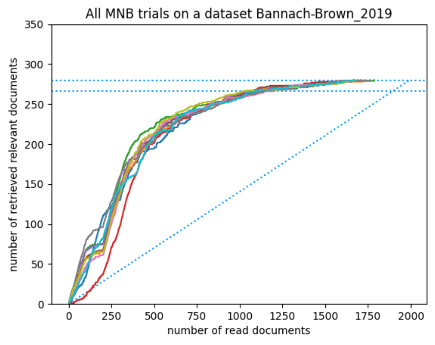


(b) Multinomial Naive Bayes

FIGURE B.1: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Appenzeller-Herzog_2020.

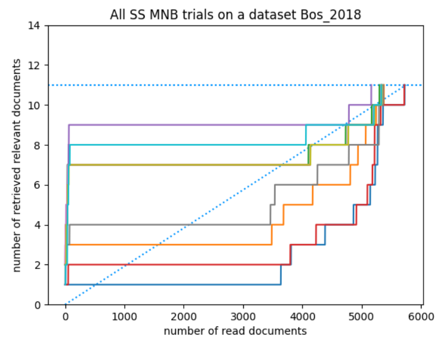


(a) Semi-supervised Multinomial Naive Bayes

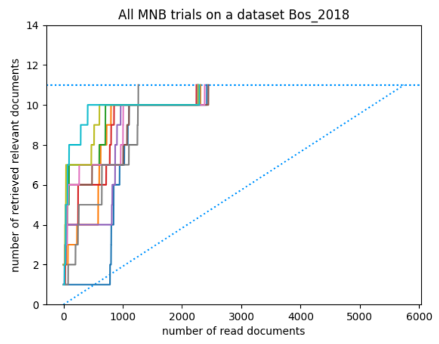


(b) Multinomial Naive Bayes

FIGURE B.2: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Bannach-Brown_2019.

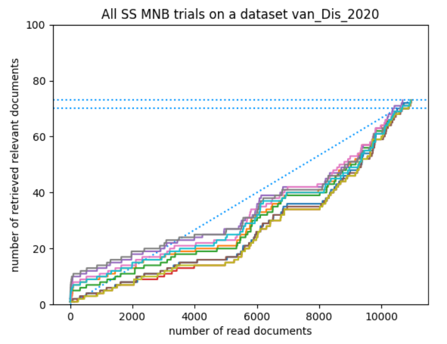


(a) Semi-supervised Multinomial Naive Bayes

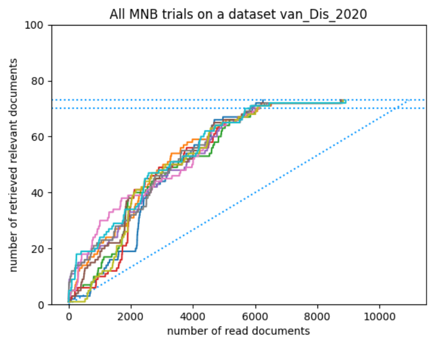


(b) Multinomial Naive Bayes

FIGURE B.3: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Bos_2018.

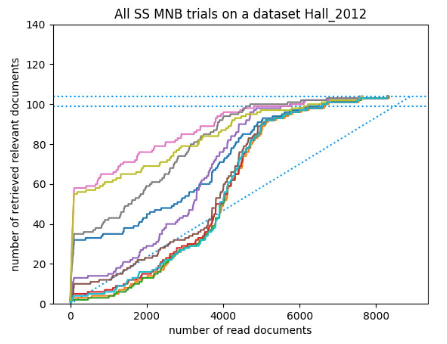


(a) Semi-supervised Multinomial Naive Bayes

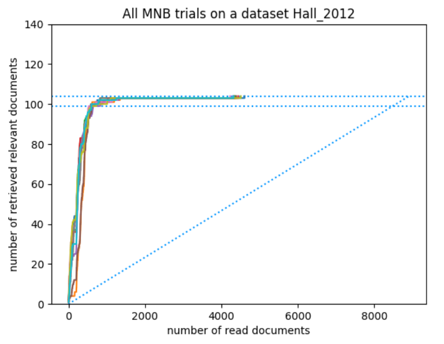


(b) Multinomial Naive Bayes

FIGURE B.4: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset van_Dis_2020.

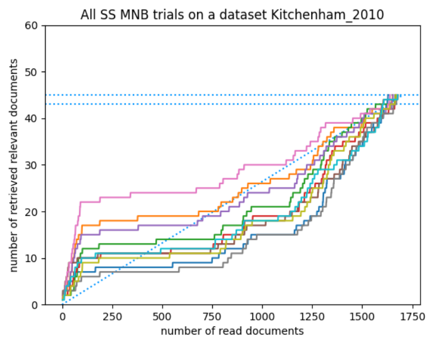


(a) Semi-supervised Multinomial Naive Bayes

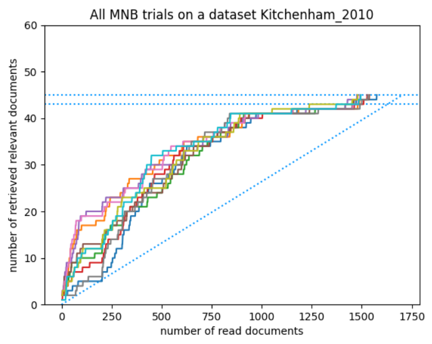


(b) Multinomial Naive Bayes

FIGURE B.5: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Hall_2012.

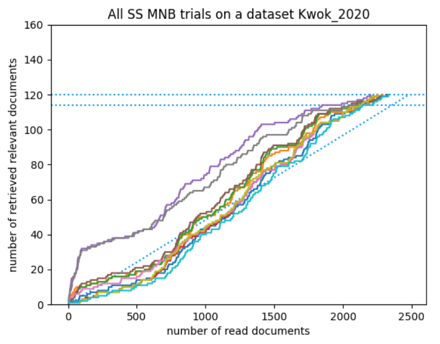


(a) Semi-supervised Multinomial Naive Bayes

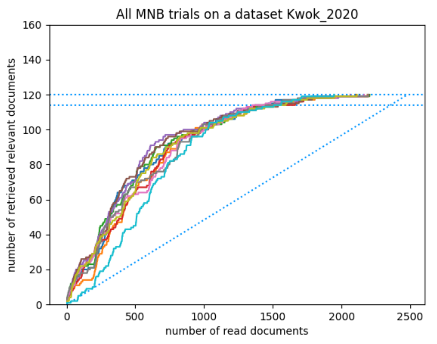


(b) Multinomial Naive Bayes

FIGURE B.6: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Kitchenham_2010.

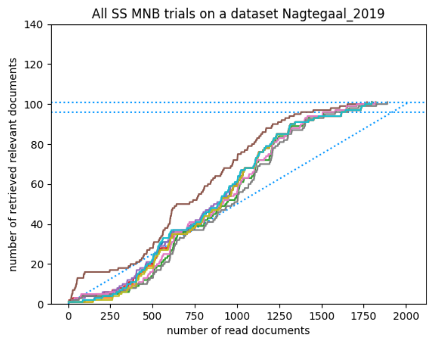


(a) Semi-supervised Multinomial Naive Bayes

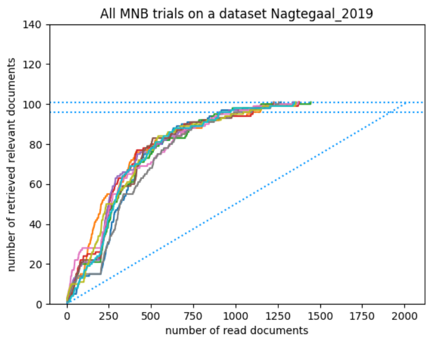


(b) Multinomial Naive Bayes

FIGURE B.7: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Kwok_2020.

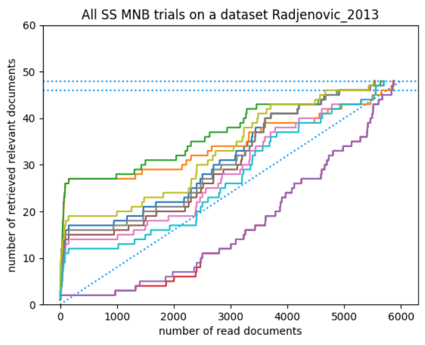


(a) Semi-supervised Multinomial Naive Bayes

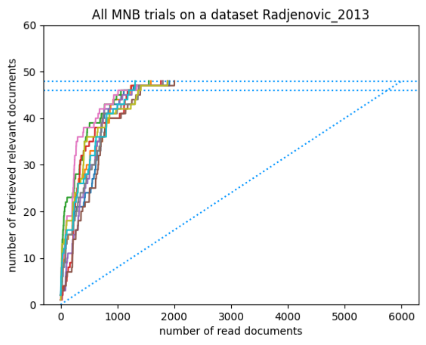


(b) Multinomial Naive Bayes

FIGURE B.8: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Nagtegaal_2019.

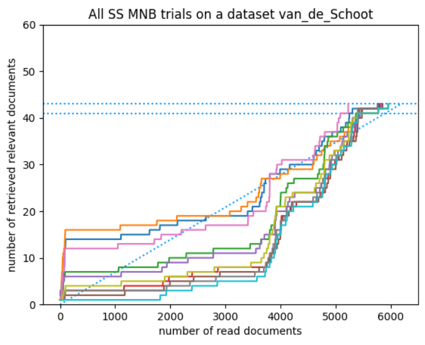


(a) Semi-supervised Multinomial Naive Bayes

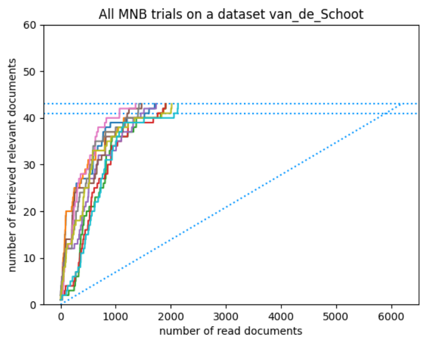


(b) Multinomial Naive Bayes

FIGURE B.9: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Radjenovic_2013.

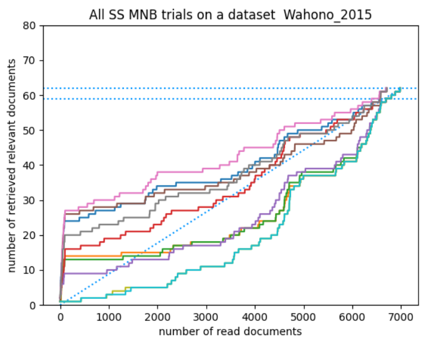


(a) Semi-supervised Multinomial Naive Bayes

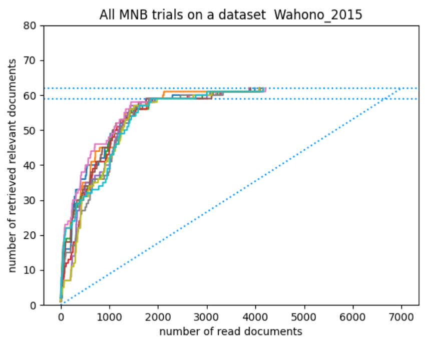


(b) Multinomial Naive Bayes

FIGURE B.10: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset van_de_Schoot.

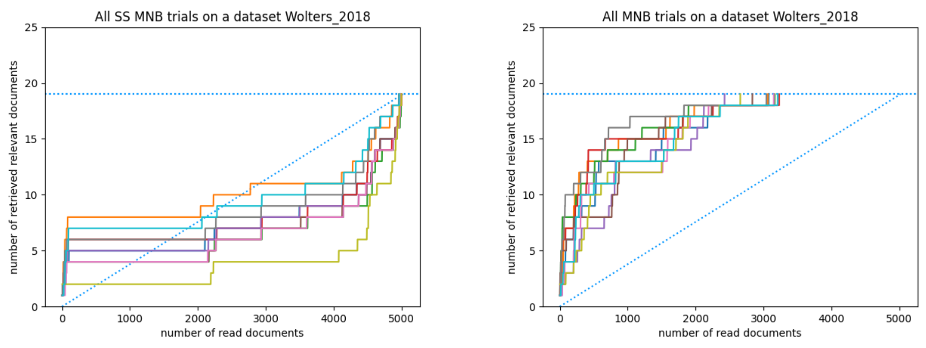


(a) Semi-supervised Multinomial Naive Bayes



(b) Multinomial Naive Bayes

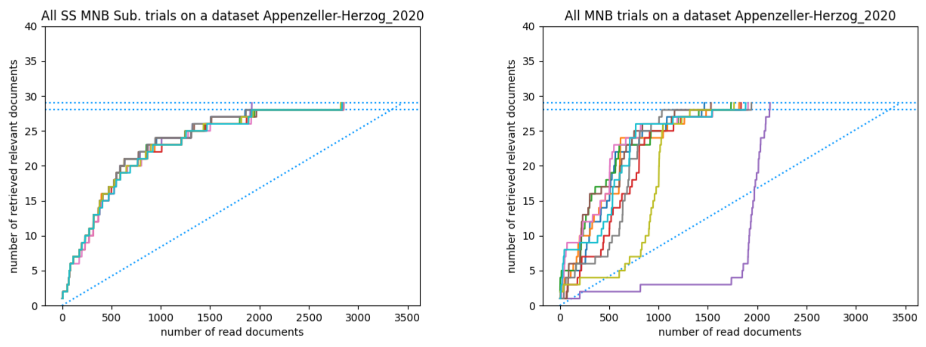
FIGURE B.11: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Wahono_2015.



(a) Semi-supervised Multinomial Naive Bayes (b) Multinomial Naive Bayes

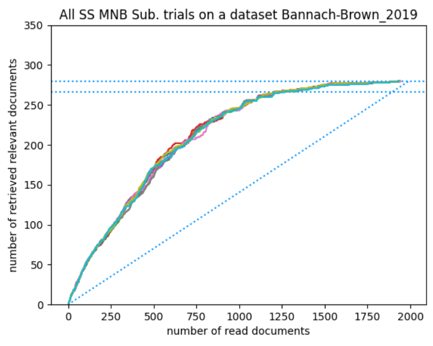
FIGURE B.12: All trials for semi-supervised Multinomial Naive Bayes (a) and Multinomial Naive Bayes (b) for the dataset Wolters_2018.

B.2 Experiment 2: Semi-supervised Multinomial Naive Bayes with sub-topics and Multinomial Naive Bayes models

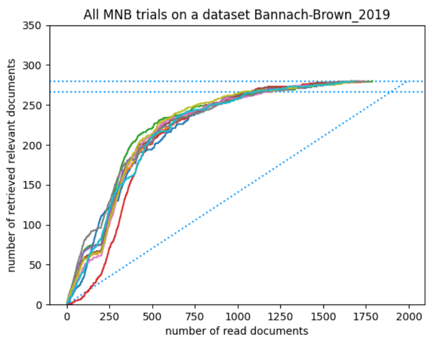


(a) Semi-supervised Multinomial Naive Bayes with sub-topics (b) Multinomial Naive Bayes

FIGURE B.13: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Appenzeller-Herzog_2020.

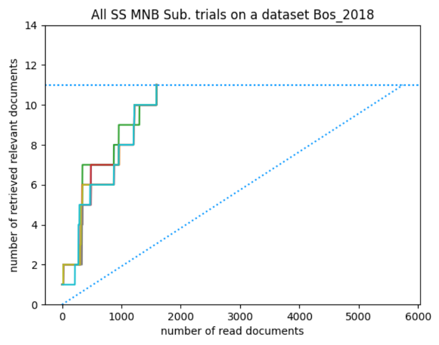


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

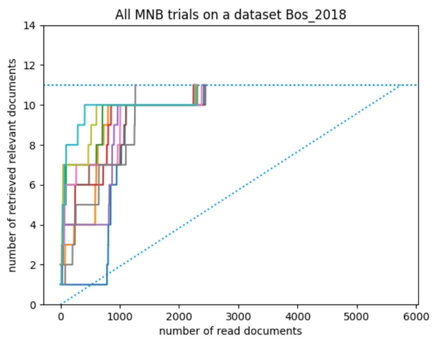


(b) Multinomial Naive Bayes

FIGURE B.14: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Bannach-Brown_2019.

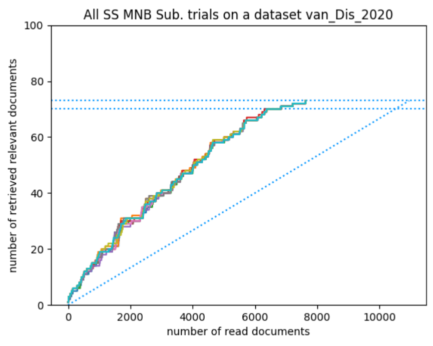


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

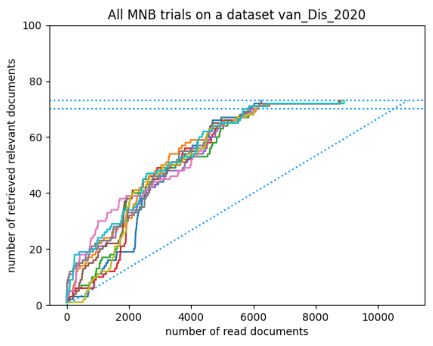


(b) Multinomial Naive Bayes

FIGURE B.15: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Bos_2018.

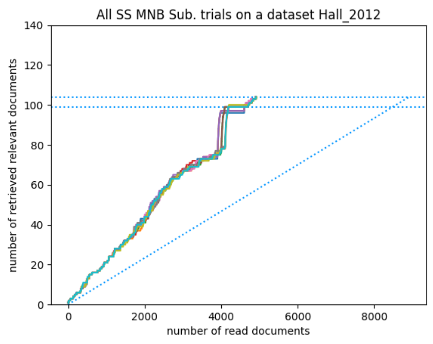


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

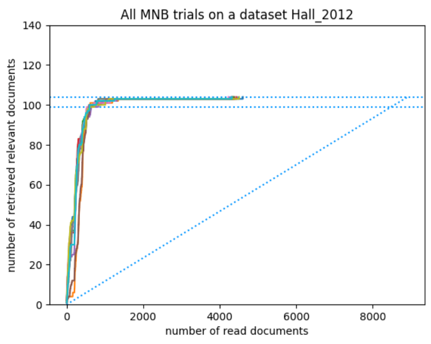


(b) Multinomial Naive Bayes

FIGURE B.16: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset van_Dis_2020.

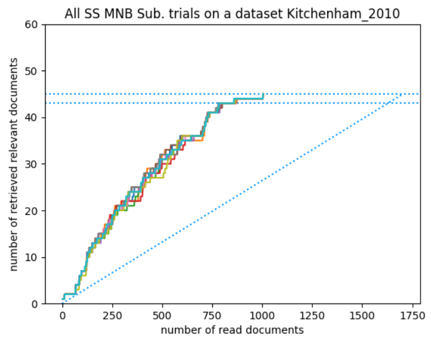


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

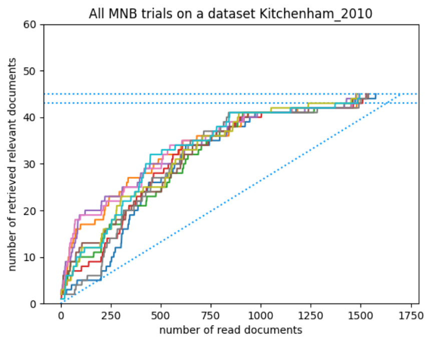


(b) Multinomial Naive Bayes

FIGURE B.17: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Hall_2012.

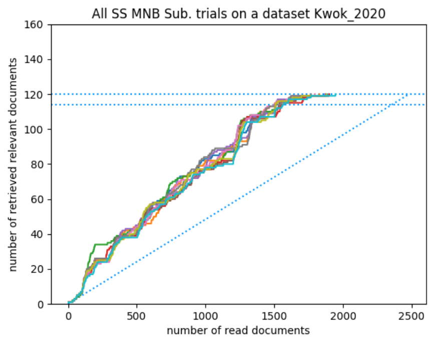


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

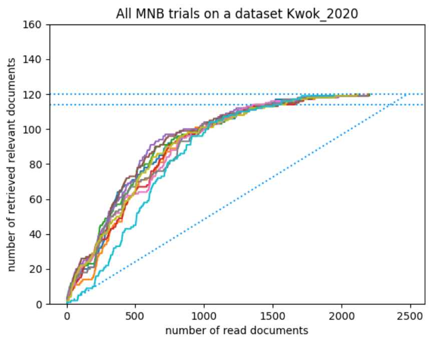


(b) Multinomial Naive Bayes

FIGURE B.18: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Kitchenham_2010.

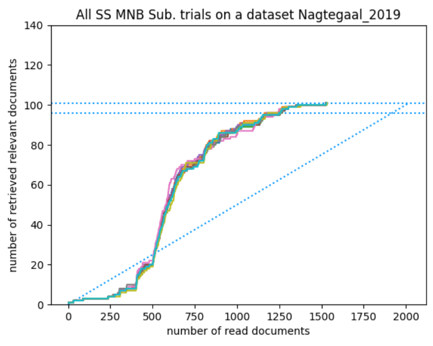


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

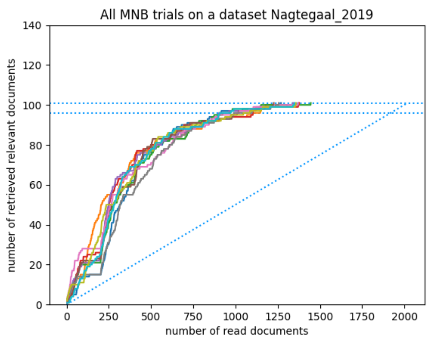


(b) Multinomial Naive Bayes

FIGURE B.19: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Kwok_2020.

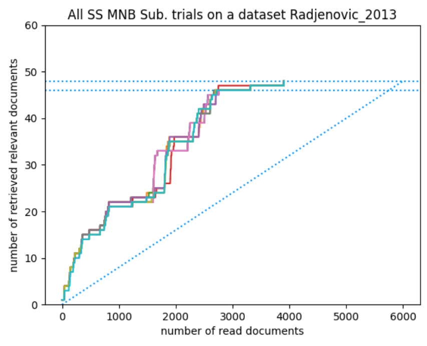


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

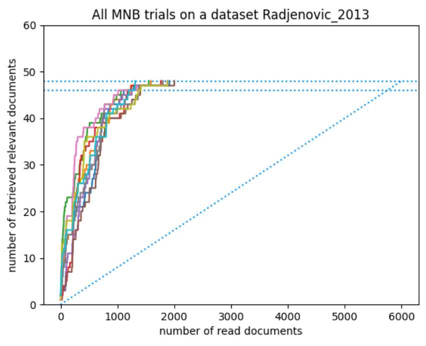


(b) Multinomial Naive Bayes

FIGURE B.20: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Nagtegaal_2019.

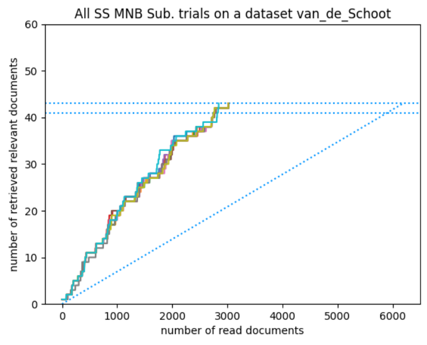


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

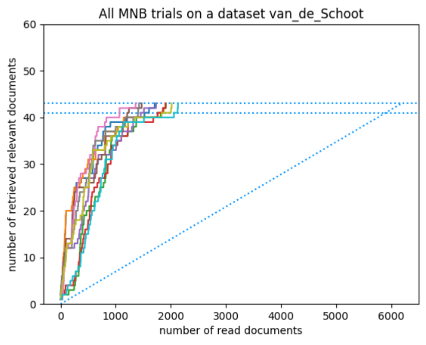


(b) Multinomial Naive Bayes

FIGURE B.21: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Radjenovic_2013.

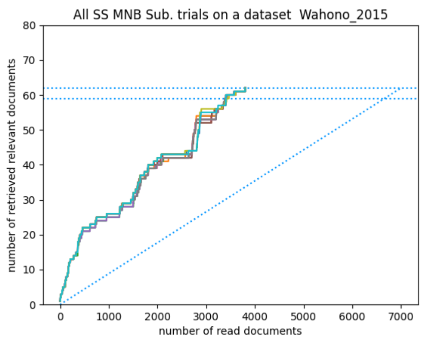


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

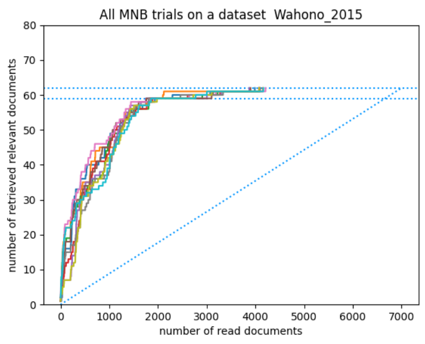


(b) Multinomial Naive Bayes

FIGURE B.22: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset van_de_Schoot.

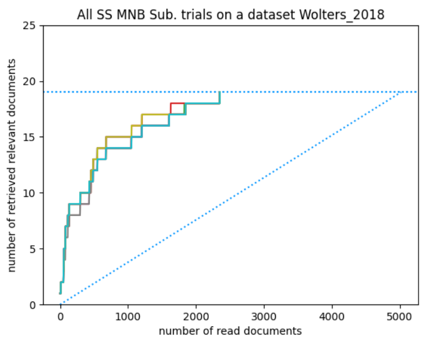


(a) Semi-supervised Multinomial Naive Bayes with sub-topics

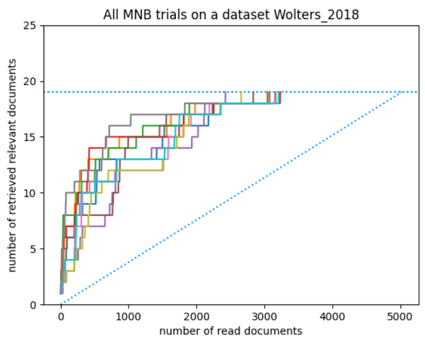


(b) Multinomial Naive Bayes

FIGURE B.23: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Wahono_2015.



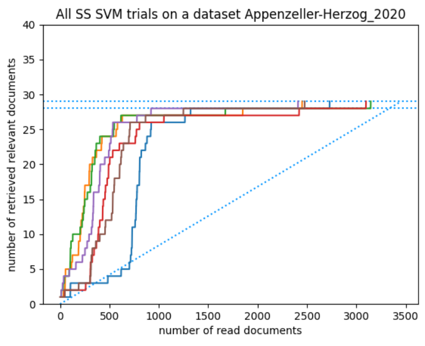
(a) Semi-supervised Multinomial Naive Bayes with sub-topics



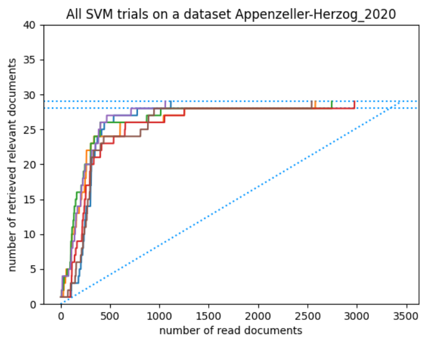
(b) Multinomial Naive Bayes

FIGURE B.24: All trials for semi-supervised Multinomial Naive Bayes with sub-topics (a) and Multinomial Naive Bayes (b) for the dataset Wolters_2018.

B.3 Experiment 3: SVM with self-training and SVM models

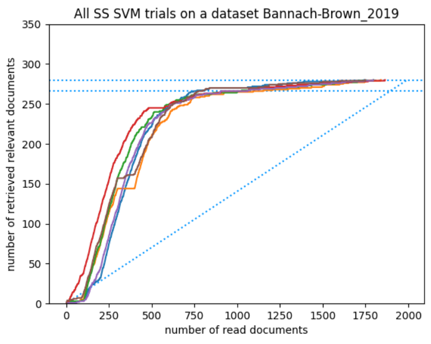


(a) SVM with self-training

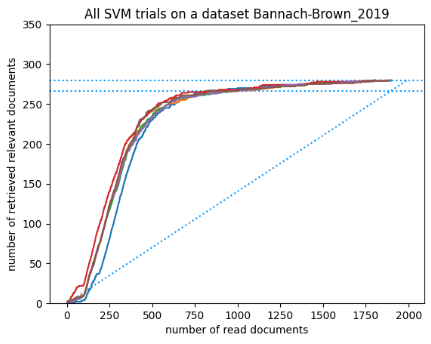


(b) SVM

FIGURE B.25: All trials for SVM with self-training (a) and SVM (b) for the dataset Appenzeller-Herzog_2020.

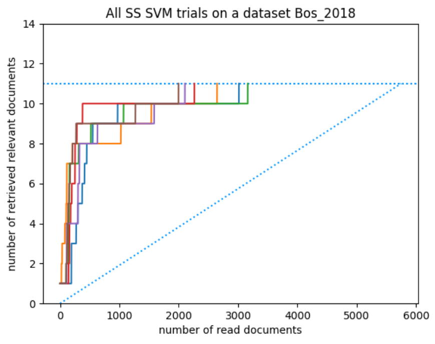


(a) SVM with self-training

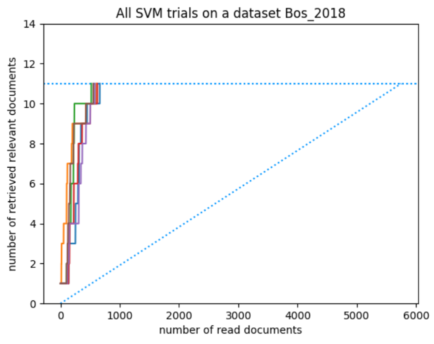


(b) SVM

FIGURE B.26: All trials for SVM with self-training (a) and SVM (b) for the dataset Bannach-Brown_2019.

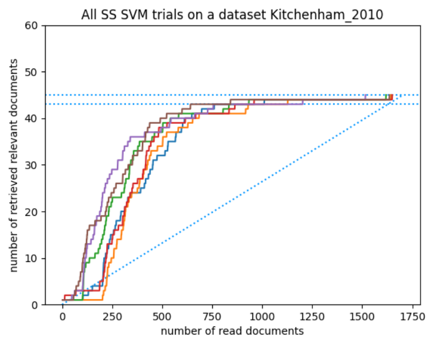


(a) SVM with self-training

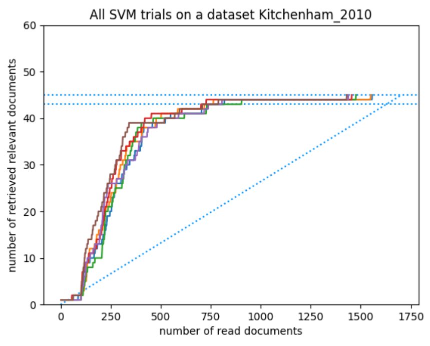


(b) SVM

FIGURE B.27: All trials for SVM with self-training (a) and SVM (b) for the dataset Bos_2018.

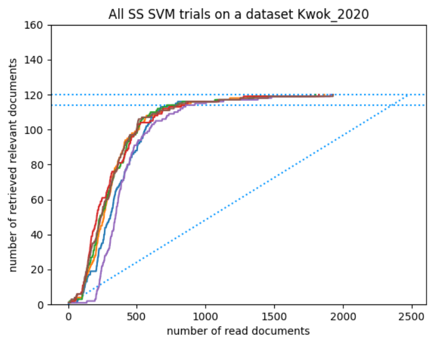


(a) SVM with self-training

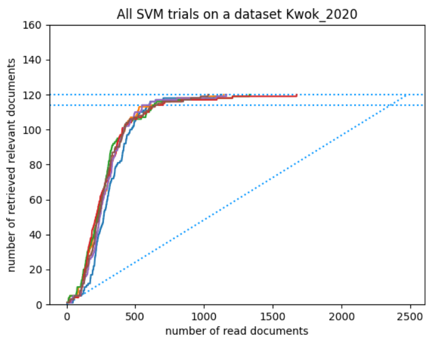


(b) SVM

FIGURE B.28: All trials for SVM with self-training (a) and SVM (b) for the dataset Kitchenham_2010.

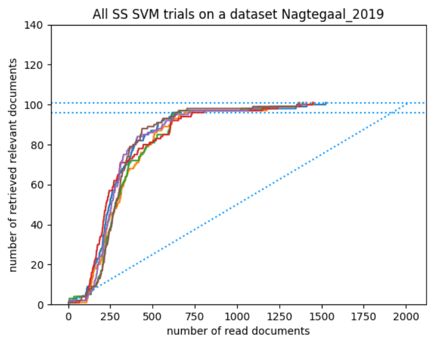


(a) SVM with self-training

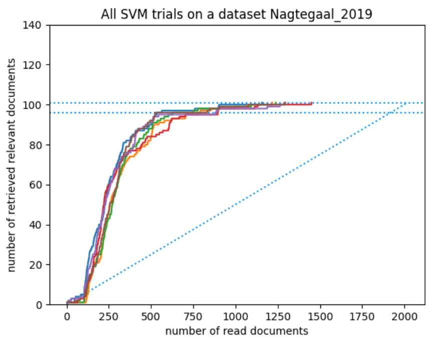


(b) SVM

FIGURE B.29: All trials for SVM with self-training (a) and SVM (b) for the dataset Kwok_2020.

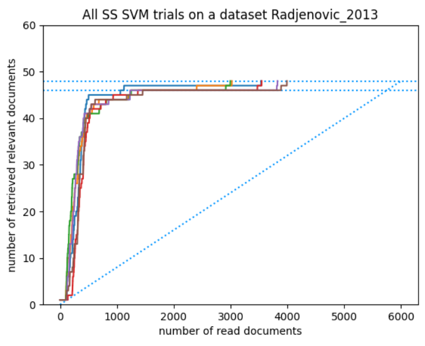


(a) SVM with self-training

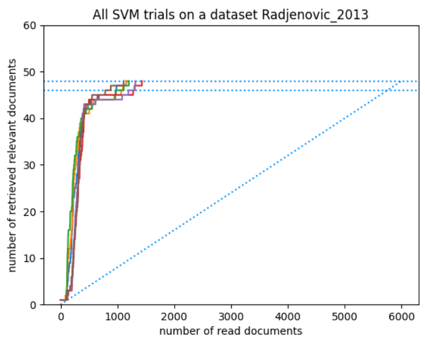


(b) SVM

FIGURE B.30: All trials for SVM with self-training (a) and SVM (b) for the dataset Nagtegaal_2019.

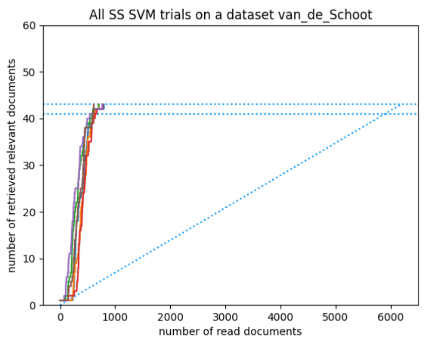


(a) SVM with self-training

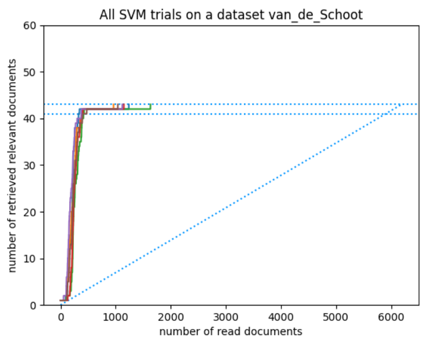


(b) SVM

FIGURE B.31: All trials for SVM with self-training (a) and SVM (b) for the dataset Radjenovic_2013.

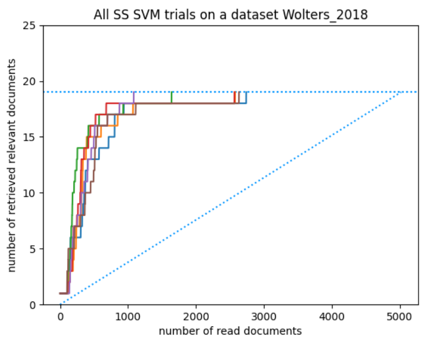


(a) SVM with self-training

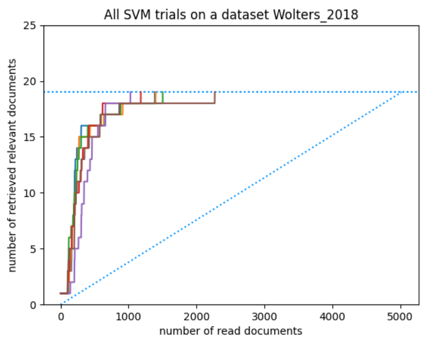


(b) SVM

FIGURE B.32: All trials for SVM with self-training (a) and SVM (b) for the dataset van_de_Schoot.



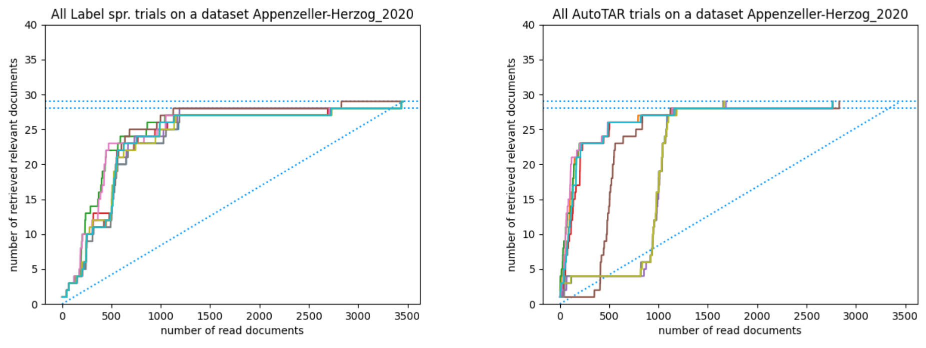
(a) SVM with self-training



(b) SVM

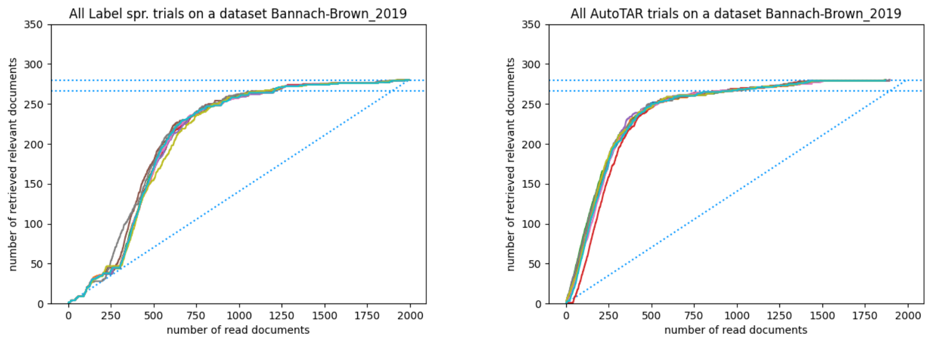
FIGURE B.33: All trials for SVM with self-training (a) and SVM (b) for the dataset Wolters_2018.

B.4 Experiment 4: Label spreading and AutoTAR models



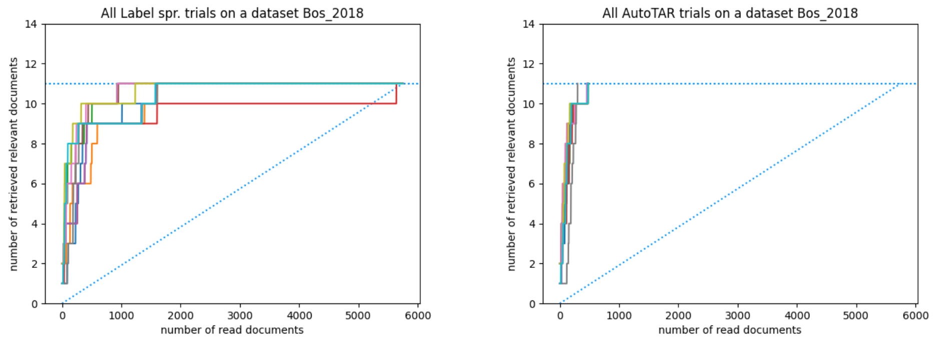
(a) Label spreading (b) AutoTAR

FIGURE B.34: All trials for Label spreading (a) and AutoTAR (b) for the dataset Appenzeller-Herzog_2020.



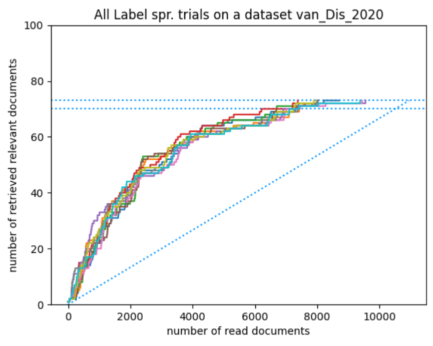
(a) Label spreading (b) AutoTAR

FIGURE B.35: All trials for Label spreading (a) and AutoTAR (b) for the dataset Bannach-Brown_2019.

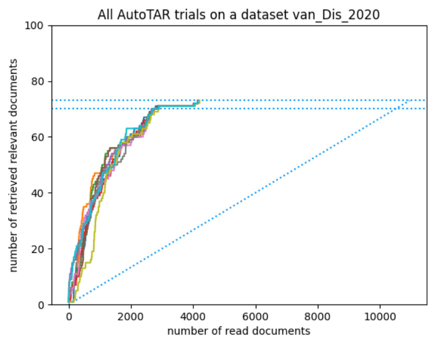


(a) Label spreading (b) AutoTAR

FIGURE B.36: All trials for Label spreading (a) and AutoTAR (b) for the dataset Bos_2018.

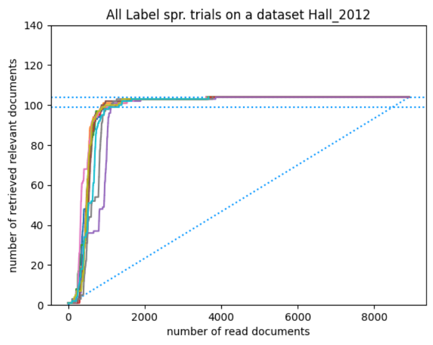


(a) Label spreading

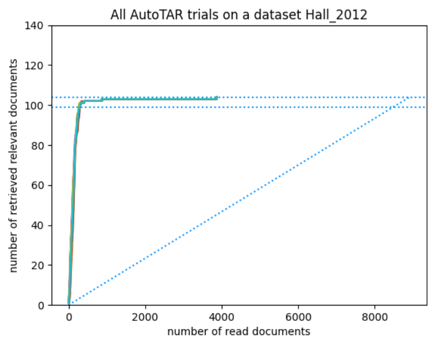


(b) AutoTAR

FIGURE B.37: All trials for Label spreading (a) and AutoTAR (b) for the dataset van_Dis_2020.

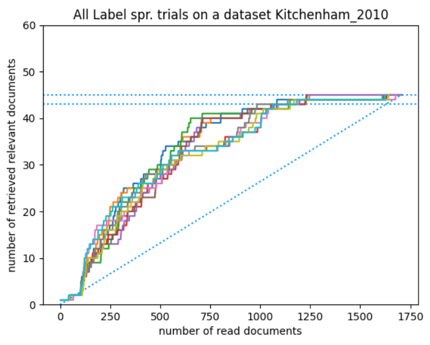


(a) Label spreading

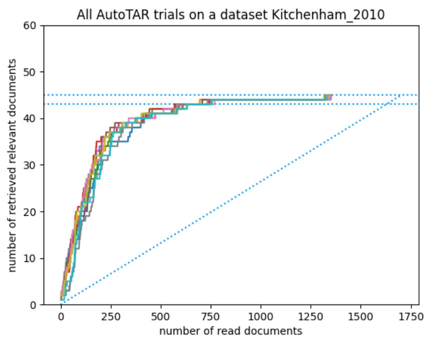


(b) AutoTAR

FIGURE B.38: All trials for Label spreading (a) and AutoTAR (b) for the dataset Hall_2012.

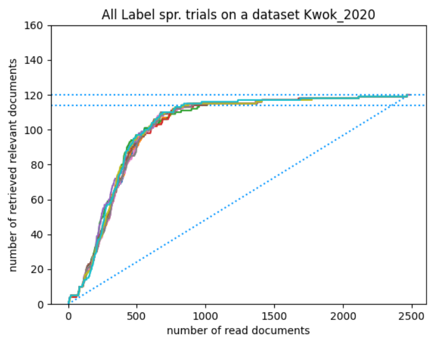


(a) Label spreading

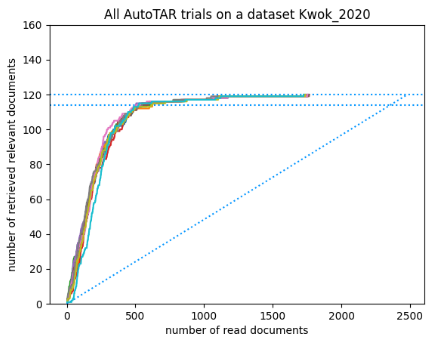


(b) AutoTAR

FIGURE B.39: All trials for Label spreading (a) and AutoTAR (b) for the dataset Kitchenham_2010.

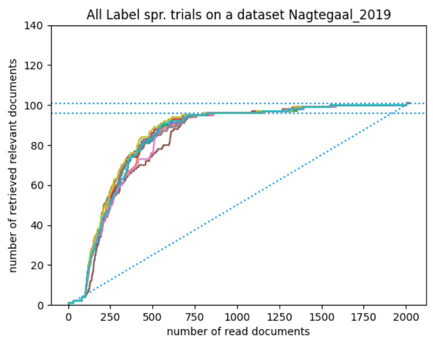


(a) Label spreading

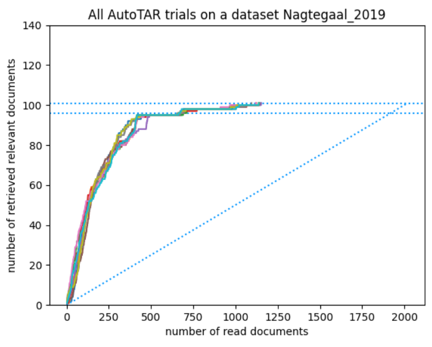


(b) AutoTAR

FIGURE B.40: All trials for Label spreading (a) and AutoTAR (b) for the dataset Kwok_2020.

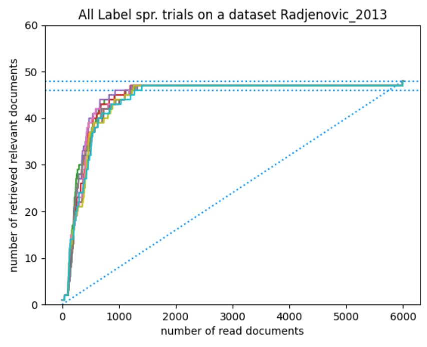


(a) Label spreading

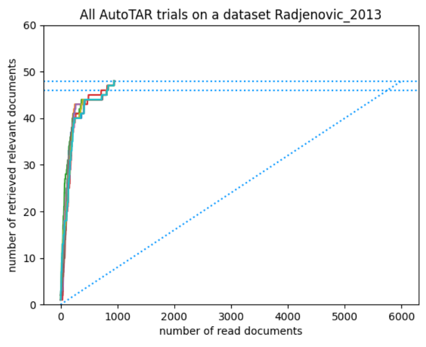


(b) AutoTAR

FIGURE B.41: All trials for Label spreading (a) and AutoTAR (b) for the dataset Nagtegaal_2019.

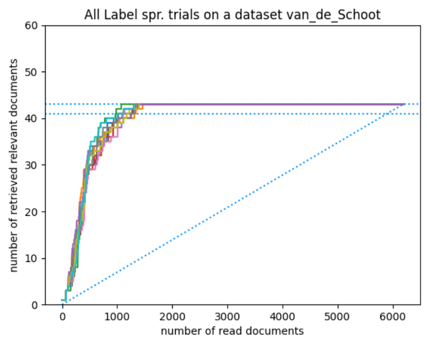


(a) Label spreading

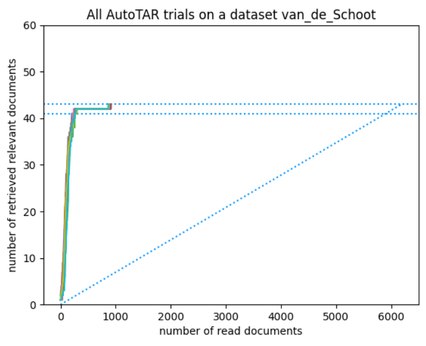


(b) AutoTAR

FIGURE B.42: All trials for Label spreading (a) and AutoTAR (b) for the dataset Radjenovic_2013.

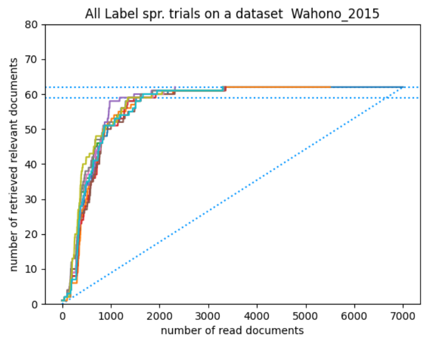


(a) Label spreading

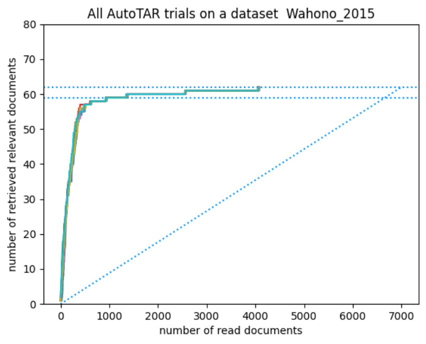


(b) AutoTAR

FIGURE B.43: All trials for Label spreading (a) and AutoTAR (b) for the dataset van_de_Schoot.

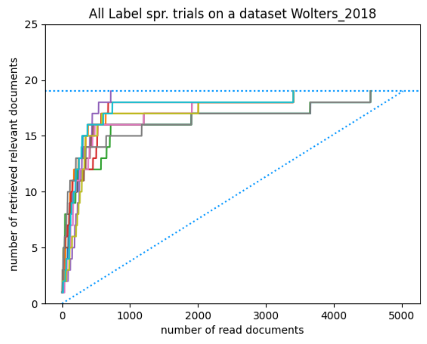


(a) Label spreading

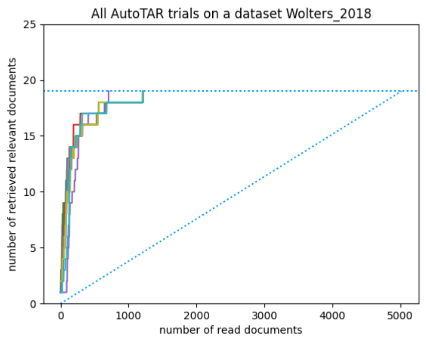


(b) AutoTAR

FIGURE B.44: All trials for Label spreading (a) and AutoTAR (b) for the dataset Wahono_2015.



(a) Label spreading

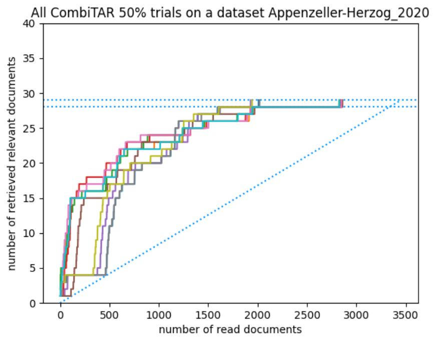


(b) AutoTAR

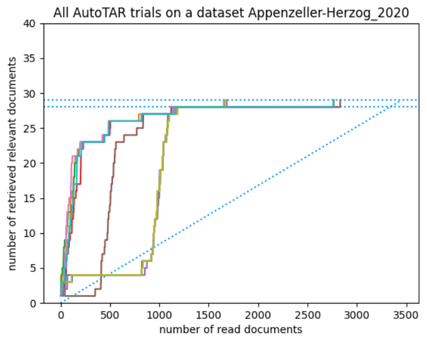
FIGURE B.45: All trials for Label spreading (a) and AutoTAR (b) for the dataset Wolters_2018.

B.5 Experiment 5: CombiTAR and AutoTAR models

B.5.1 CombiTAR 50% and AutoTAR models

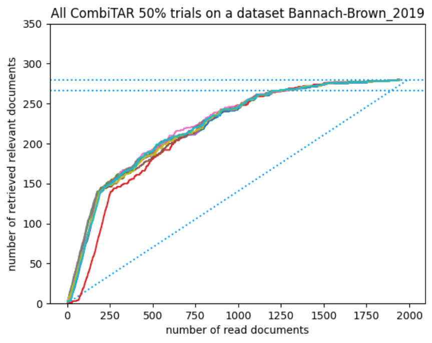


(a) CombiTAR 50%

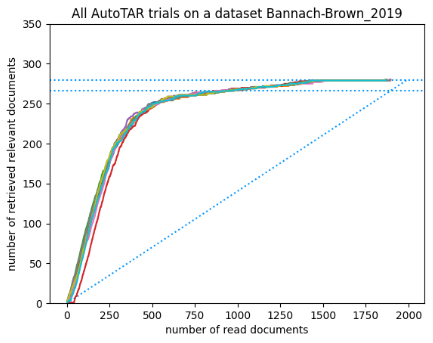


(b) AutoTAR

FIGURE B.46: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Appenzeller-Herzog_2020.

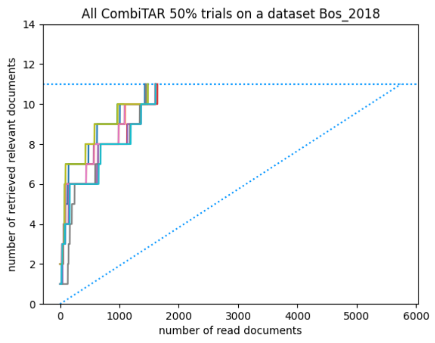


(a) CombiTAR 50%

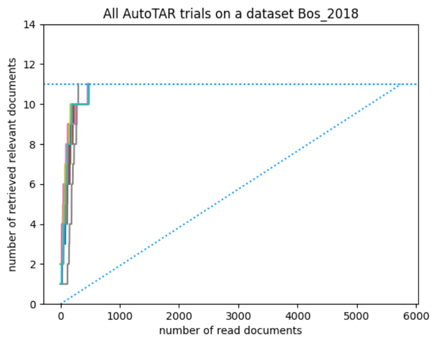


(b) AutoTAR

FIGURE B.47: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Bannach-Brown_2019.

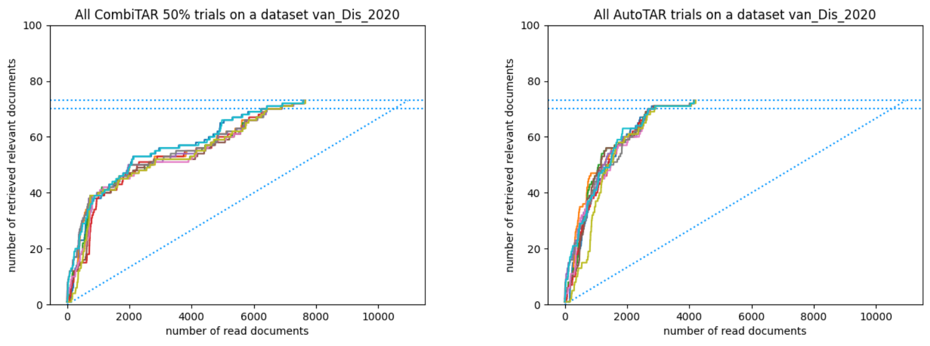


(a) CombiTAR 50%



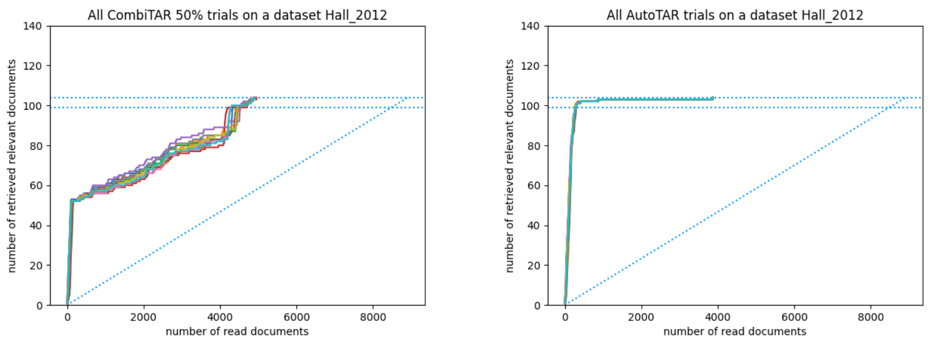
(b) AutoTAR

FIGURE B.48: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Bos_2018.



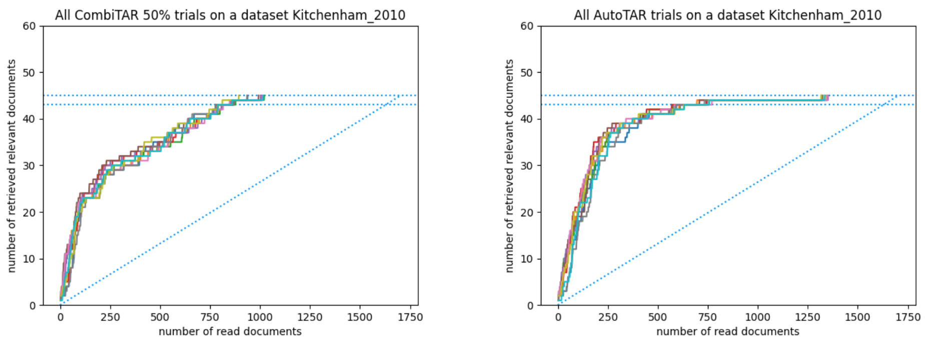
(a) CombiTAR 50% (b) AutoTAR

FIGURE B.49: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset van_Dis_2020.



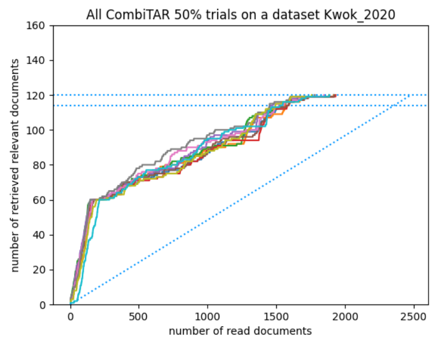
(a) CombiTAR 50% (b) AutoTAR

FIGURE B.50: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Hall_2012.

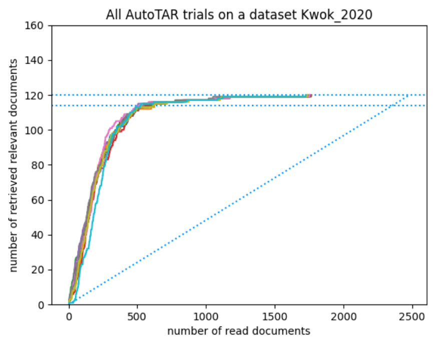


(a) CombiTAR 50% (b) AutoTAR

FIGURE B.51: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Kitchenham_2010.

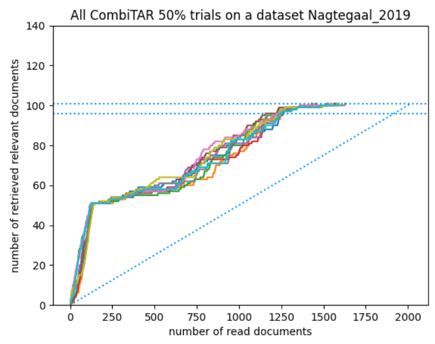


(a) CombiTAR 50%

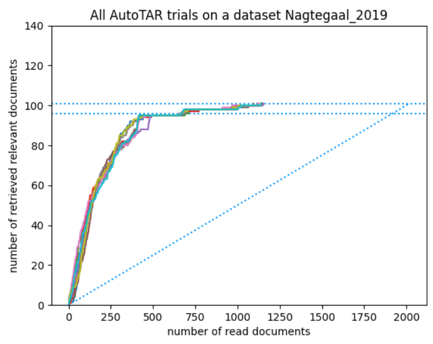


(b) AutoTAR

FIGURE B.52: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Kwok_2020.

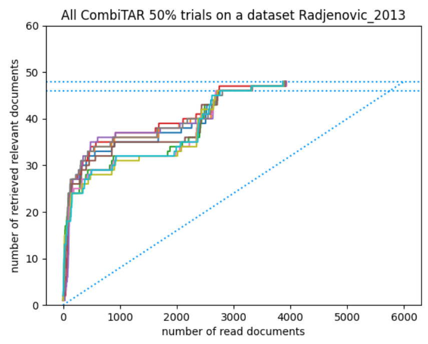


(a) CombiTAR 50%

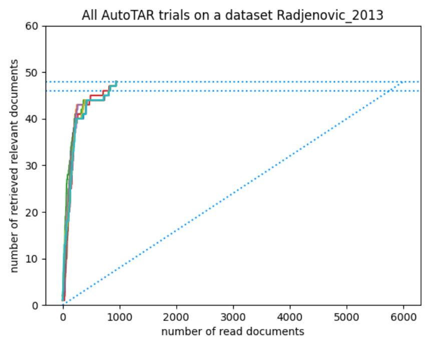


(b) AutoTAR

FIGURE B.53: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Nagtegaal_2019.

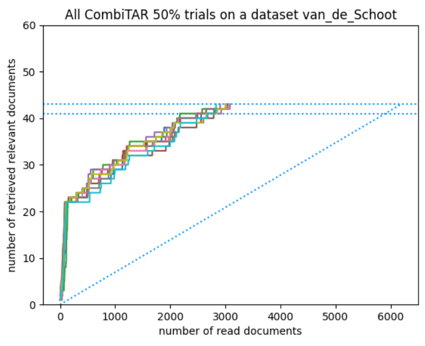


(a) CombiTAR 50%

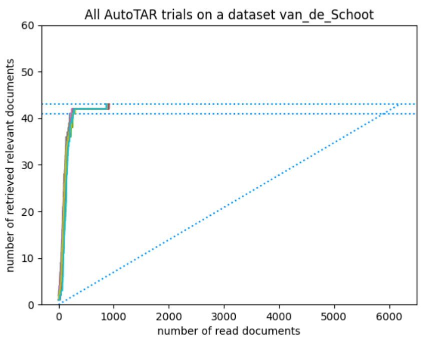


(b) AutoTAR

FIGURE B.54: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Radjenovic_2013.

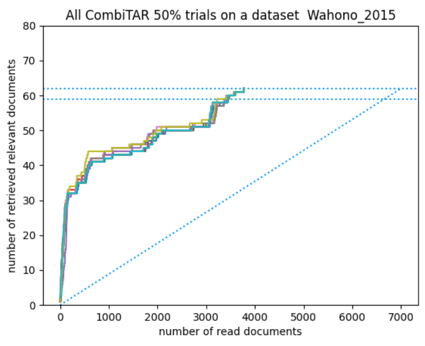


(a) CombiTAR 50%

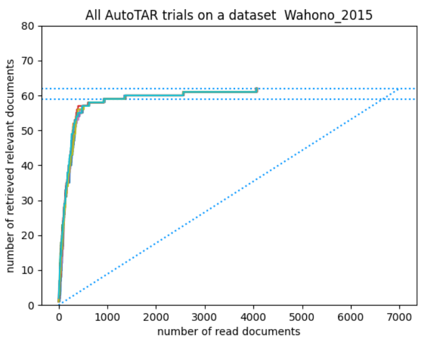


(b) AutoTAR

FIGURE B.55: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset van_de_Schoot.

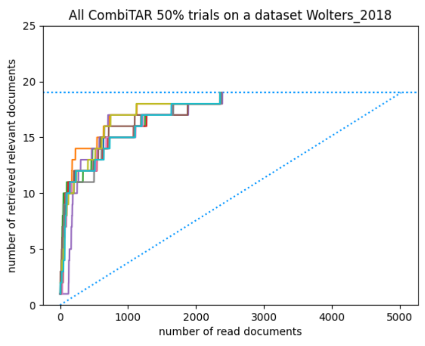


(a) CombiTAR 50%

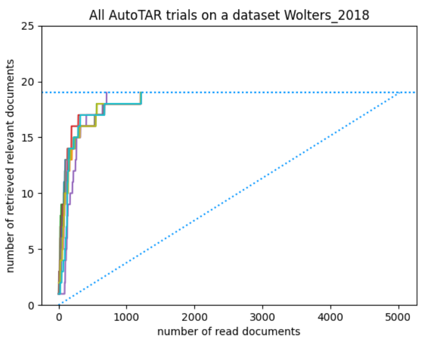


(b) AutoTAR

FIGURE B.56: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Wahono_2015.



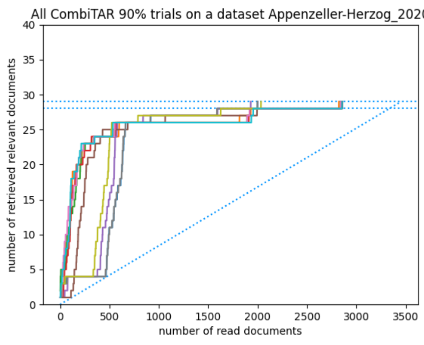
(a) CombiTAR 50%



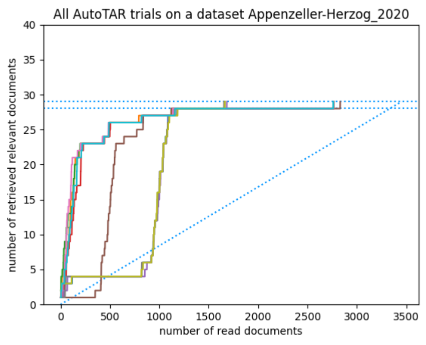
(b) AutoTAR

FIGURE B.57: All trials for CombiTAR 50% (a) and AutoTAR (b) for the dataset Wolters_2018.

B.5.2 CombiTAR 90% and AutoTAR models

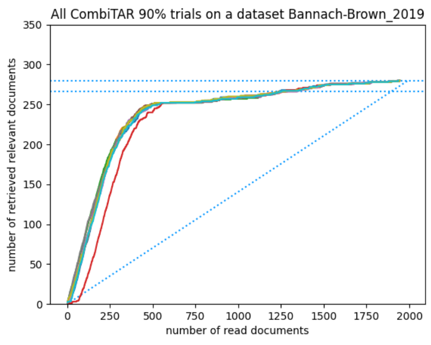


(a) CombiTAR 90%

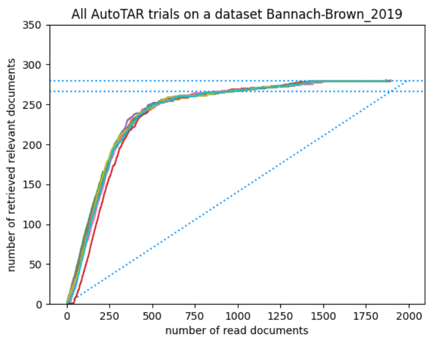


(b) AutoTAR

FIGURE B.58: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Appenzeller-Herzog_2020.

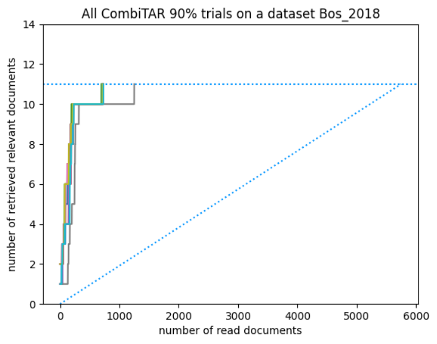


(a) CombiTAR 90%

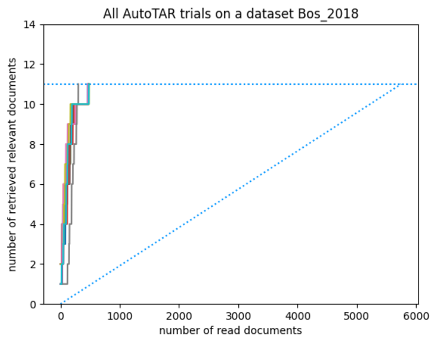


(b) AutoTAR

FIGURE B.59: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Bannach-Brown_2019.

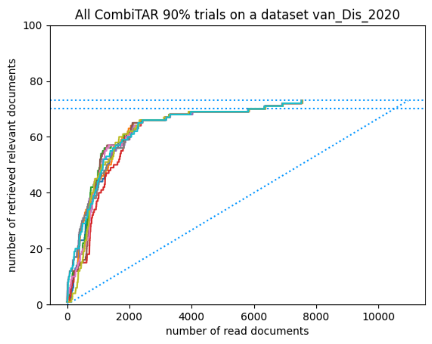


(a) CombiTAR 90%

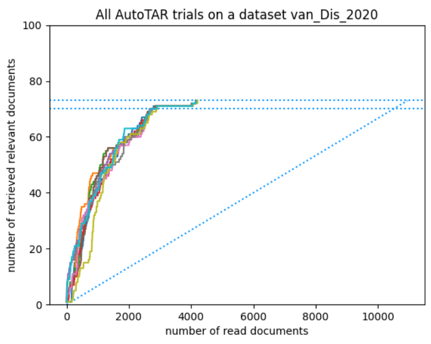


(b) AutoTAR

FIGURE B.60: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Bos_2018.

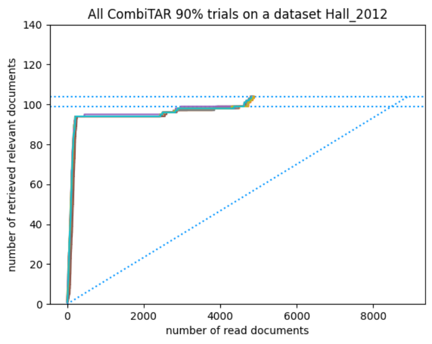


(a) CombiTAR 90%

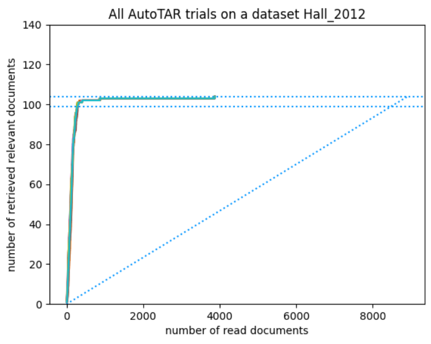


(b) AutoTAR

FIGURE B.61: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset van_Dis_2020.

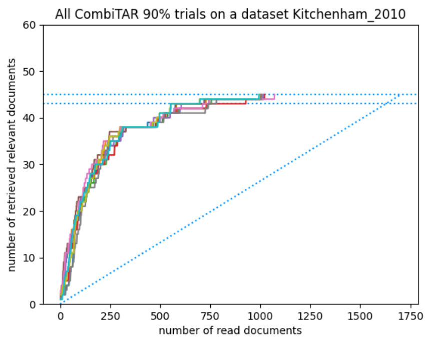


(a) CombiTAR 90%

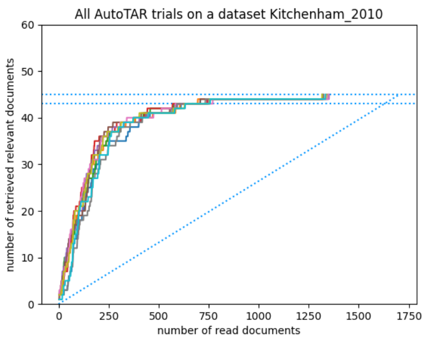


(b) AutoTAR

FIGURE B.62: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Hall_2012.

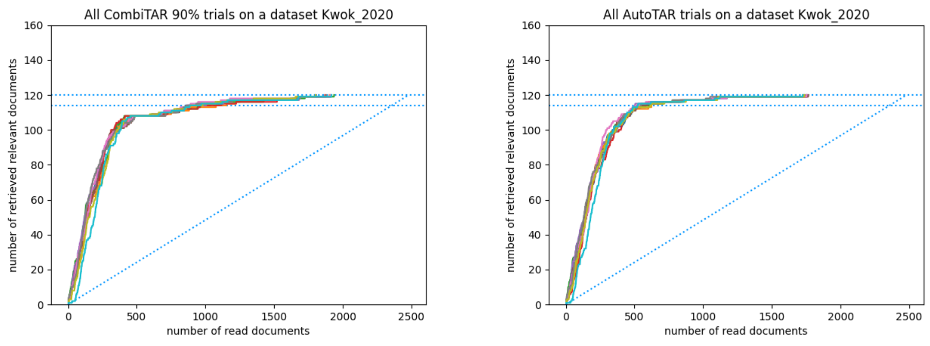


(a) CombiTAR 90%



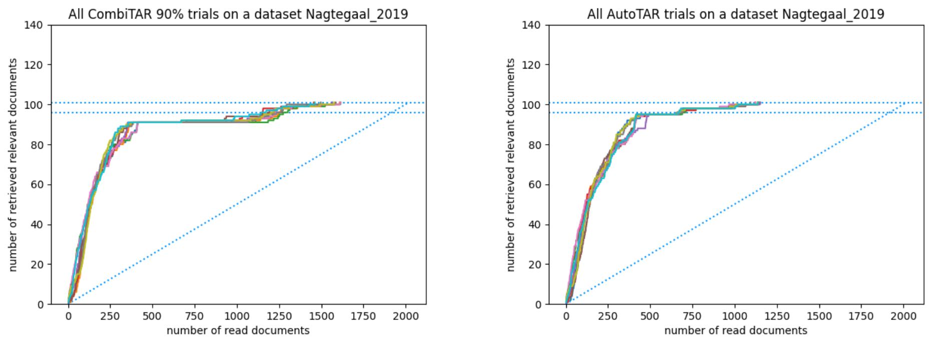
(b) AutoTAR

FIGURE B.63: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Kitchenham_2010.



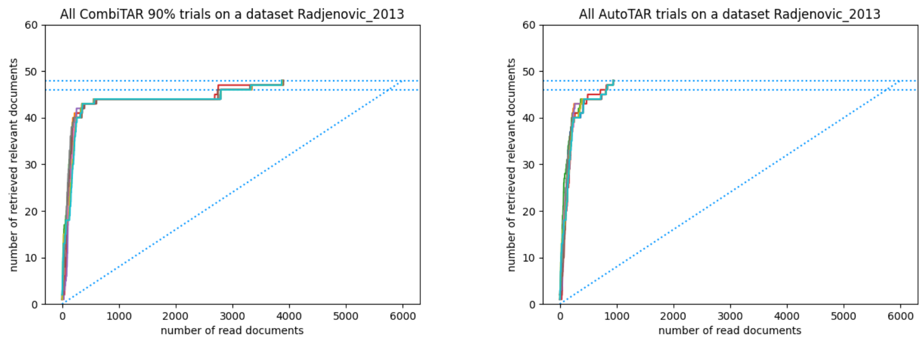
(a) CombiTAR 90%   (b) AutoTAR

FIGURE B.64: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Kwok_2020.



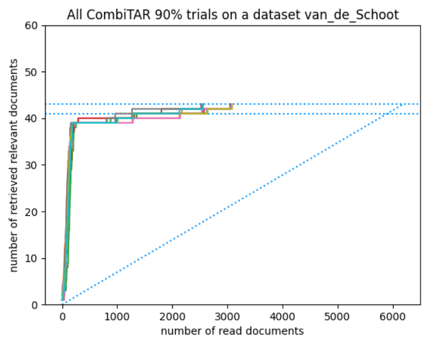
(a) CombiTAR 90%   (b) AutoTAR

FIGURE B.65: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Nagtegaal_2019.

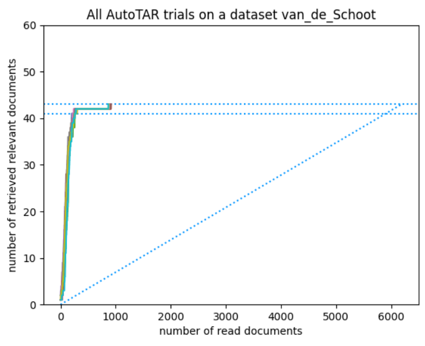


(a) CombiTAR 90%   (b) AutoTAR

FIGURE B.66: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Radjenovic_2013.

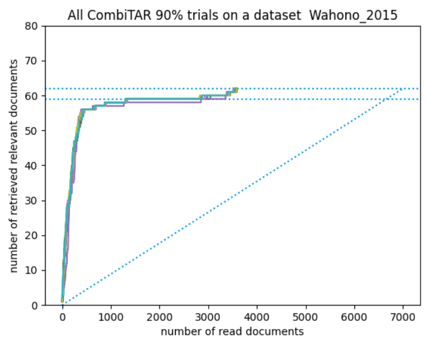


(a) CombiTAR 90%

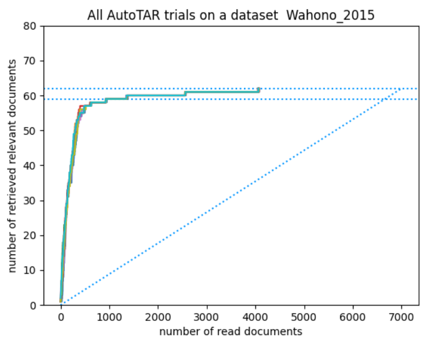


(b) AutoTAR

FIGURE B.67: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset van_de_Schoot.

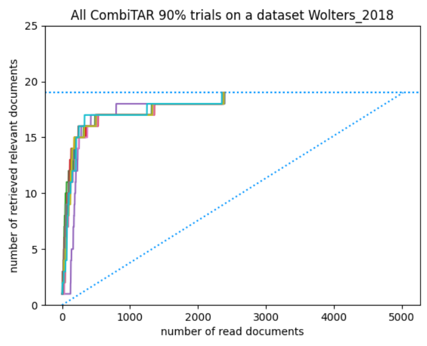


(a) CombiTAR 90%

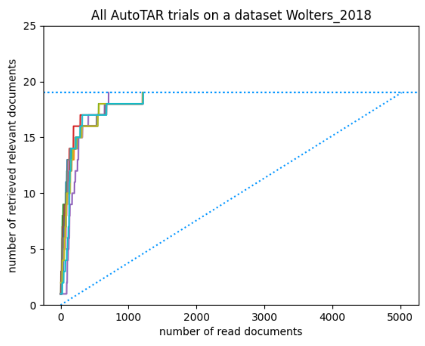


(b) AutoTAR

FIGURE B.68: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Wahono_2015.



(a) CombiTAR 90%



(b) AutoTAR

FIGURE B.69: All trials for CombiTAR 90% (a) and AutoTAR (b) for the dataset Wolters_2018.

Appendix C

Ethics and Privacy Quick Scan

Response Summary:

Section 1. Research projects involving human participants

P1. Does your project involve human participants? This includes for example use of observation, (online) surveys, interviews, tests, focus groups, and workshops where human participants provide information or data to inform the research. If you are only using existing data sets or publicly available data (e.g. from Twitter, Reddit) without directly recruiting participants, please answer no.

- No

Section 2. Data protection, handling, and storage

The General Data Protection Regulation imposes several obligations for the use of **personal data** (defined as any information relating to an identified or identifiable living person) or including the use of personal data in research.

D1. Are you gathering or using personal data (defined as any information relating to an identified or identifiable living person)?

- No

Section 3. Research that may cause harm

Research may cause harm to participants, researchers, the university, or society. This includes when technology has dual-use, and you investigate an innocent use, but your results could be used by others in a harmful way. If you are unsure regarding possible harm to the university or society, please discuss your concerns with the Research Support Office.

H1. Does your project give rise to a realistic risk to the national security of any country?

- No

H2. Does your project give rise to a realistic risk of aiding human rights abuses in any country?

- No

H3. Does your project (and its data) give rise to a realistic risk of damaging the University's reputation? (E.g., bad press coverage, public protest.)

- No

H4. Does your project (and in particular its data) give rise to an increased risk of attack (cyber- or otherwise) against the University? (E.g., from pressure groups.)

- No

H5. Is the data likely to contain material that is indecent, offensive, defamatory, threatening, discriminatory, or extremist?

- No

H6. Does your project give rise to a realistic risk of harm to the researchers?

- No

H7. Is there a realistic risk of any participant experiencing physical or psychological harm or discomfort?

- No

H8. Is there a realistic risk of any participant experiencing a detriment to their interests as a result of participation?

- No

H9. Is there a realistic risk of other types of negative externalities?

- No

Section 4. Conflicts of interest

C1. Is there any potential conflict of interest (e.g. between research funder and researchers or participants and researchers) that may potentially affect the research outcome or the dissemination of research findings?

- No

C2. Is there a direct hierarchical relationship between researchers and participants?

- No

Section 5. Your information.

This last section collects data about you and your project so that we can register that you completed the Ethics and Privacy Quick Scan, sent you (and your supervisor/course coordinator) a summary of what you filled out, and follow up where a fuller ethics review and/or privacy assessment is needed. For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information. Please see the guidance on the ICS Ethics and Privacy website on what happens on submission.

Z0. Which is your main department?

- Information and Computing Science

Z1. Your full name:
Ercan Öz

Z2. Your email address:
e.oz@students.uu.nl

Z3. In what context will you conduct this research?

- As a student for my master thesis, supervised by::
  Dr. A.J. (Ad) Feelders

Z5. Master programme for which you are doing the thesis

- Artificial Intelligence

Z6. Email of the course coordinator or supervisor (so that we can inform them that you filled this out and provide them with a summary):
a.j.feelders@uu.nl

Z7. Email of the moderator (as provided by the coordinator of your thesis project):
g.m.krempl@uu.nl (Second Examiner)

Z8. Title of the research project/study for which you filled out this Quick Scan:
Semi-supervised learning for Technology Assisted Review

Z9. Summary of what you intend to investigate and how you will investigate this (200 words max):

Consider the task of finding all documents relevant to an information need in a (potentially large) collection of documents. To determine whether any given document is relevant is non-trivial, for example, a human has to read the document to determine its relevance. In Technology Assisted Review (TAR) one tries to speed up this process typically by using machine learning models in an active learning cycle.

Classifiers are trained on the labeled instances, and used to select the next document to be labeled from the set of unlabeled documents. In training the classifiers, the unlabeled data is typically ignored, even though semi-supervised learning techniques could potentially make use of the unlabeled data to improve the quality of the classifiers.

In this project we are going to study different techniques for semi-supervised learning (SSL) for binary classification, and evaluate the added value of these techniques in the TAR process. Since the TAR process is complex, and contains a number of different building blocks, the interaction between different SSL approaches and other TAR components must also be taken into account.

Z10. In case you encountered warnings in the survey, does supervisor already have ethical approval for a research line that fully covers your project?

- Not applicable

Scoring

- Privacy: 0
- Ethics: 0
