Manuscript drafts

Gerbrich Ferdinands

1/14/2020

Keywords: Human-Machine interaction, active learning, systematic reviews, text classification

#### Introduction

#### Methods

A convenience sample of 5 existing systematic reviews on varying topics was collected.

#### Results

#### Discussion

# Introduction

Systematic Reviews (SR’s) are booming - but they are a lot of work Various machine learning tools have been proposed to reduce workload in abstract screening.

* objectives - to demonstrate effectiveness of ml algorithm in reducing abstract classification for systematic reviews
* justification -
* background
* guidance to reader
* summary/conclusion

In building evidence in research, systematic reviews are top of the bill. A systematic review attempts to bring together all relevant studies to answer a specific research question. (PRISMA-P Group et al. 2015) Findings from systematic reviews are leading in developing clinical guidelines.

Performing a systematic review can be a tedious and time-consuming task. To gather the findings relevant to answering the research question, a systematic search is performed. A systematic search starts with collecting all publications that meet pre-specified eligibility criteria. From this collection of candidate studies the researcher has to identify the publications relevant for answering the research question.

As more and more papers are published and reproducibility crisis has emerged, A systematic search often results in thousands of candidate studies. Relevant publications are then identified by screening title and abstract of all candidate studies. This screening process is a manual task often executed by multiple reviewers to ensure reliability. This is a time consuming process that weighs heavily on resources.

A solution is …

Such a solution can save time and resources Time and resources are limited, time a researcher often does not have!! (think covid-19).

It is important reflect on research by giving an overview of research areas which is typically done by a systematic review […].

To review a specific research area, one starts out with an initial search of thousands of academic papers. All these papers abstracts need to be screened to find an initial batch of possibly relevant papers. With now hopefully only a couple of hundred papers left, the researcher needs to read these papers full-text to arrive at a final selection of papers that are relevant for the final systematic review [this is prisma process?]. This whole processes costs this and this much time [shelmilt].

Several algorithms to reduce workload in abstract screening have been proposed.

#### Proposed solution

This study is about how machine learning models can increase efficiency in systematic reviews. Their main objective is to identify ..

The strategy to reduce workload proposed in the current study is by prioritizing publications that are deemed most relevant to the systematic review. As the relevant publications are screened first, the reviewing process can be quit eaerlier.

The stage of abstract screening where abstracts are systematically screened is where a lot is to be gained. This stage is the target of possible learning algorithms that can assist the reviewer in selecting the relevant papers. Together with the reviewer /human machine interaction. The algorithm aims to compute which papers in the pool need to be excluded and which need to be included, based on the reviewers decisions. It learns from the reviewers decisions and asks the reviewer to provide more labels, incrementally improving its class predictions.

The goal of the algorithm defined in the current study is to reduce to number of abstracts needed to screen (maybe not right term, bit biomedical). To be more specific, the algorithm aims to present the reader with the primary studies as soon as possible. This means that at some point you probably have seen all relevant abstracts and are only viewing excluded papers, which means you can stop reviewing much earlier (theoretically spoken). Also reviewing is now much more fun. As compared to when you have to review all abstracts and you perhaps see only one relevant abstract every other week/day.

So you might wonder, how does such an algorithm actually work? Active learning strategy, starting with a pool of unlabeled abstracts (U). The reviewer starts labeling some instances in U, creating L. The algorithm utilizes L to predict labels for all abstracts (classifier), by using a set of features from the papers called X, for example the text in the abstract (feature extraction method). Now, it made an initial classification. The algorithm now aims to improve its classification by which paper from U will be presented to the reviewer next. By labeling the next paper, the reviewer provides the algorithm with new information which the algorithm uses to update its prediction.

We approach asr as a classification problem: All papers obtained in the systematic search form a pool of instances x. All instances x need to be classified, e.g. we want to give them a label y. all x are now part of U,. Binary classification, either inclusion or exclusion. We want to classify based on some features from the instances , feature vector .

By starting of with L, the algorithm utilizes characteristics of X to predict labels in U.

We want to classify the papers in Where the whole collection of papers in the systematic search form a pool of instances x, with unknown label y. <x,y>.

input:   
 a pool of unlabeled abstracts $\mathcal{U}$   
 oracle labels a few initial papers, from $\mathcal{U}$, $\mathcal{L}$,   
  
 M = train($\mathcal{L}$)  
 Query x \isin \mathcal{U} (<x, y=?)  
 oracle <x, y>

* pool unlabeled abstracts
* labeled data set ,
* instance , label
* utility measure
* best query instance according to

Now there are a few technical details. Many different versions of such algorithms exist. Many of such algorithms have been described in the active learning literature and have been applied in the systematic reviewing process. Not exhaustive, but the algorithm can apply many different strategies to arrive at its predictions, which can be divided in following parameters: classifier, feature extraction strategy, balancing, query strateg y.

Most often the SVM classifier is used, popular and very good results. Also lots of other configurations. However, other classifiers have not been tested a lot (polygon thing by cohe, naïve bayes and random forest by …), but mostly SVM still. Also, most research in the medical sciences (well there are some exceptions of course [conversation between cohen and matwill]

In the current study, we aim at exploring the performance of several classifiers on reducing workload while maintaining performance in abstract screening process. This is done by performing simulations on existing systematic reviews. Performance is evaluated by how much time can be saved … while maintaining accuracy.

Present research questions. RQ1 – which classifiers perform best? - RQ1a – does classifier performance vary over different research areas? (in what terms does it perform best)

RQ2 different hyperparameter optimizations? Classifiers come with hyperparameters. We have to make a choice on how to set these hyperparameters. What we do is create 3 sets, optimizing in three ways, aggregate results obtained.

The goal is to gain insight in classifiers other than the widely applied SVM, overall various research areas. So not only medical sciences.

To perform all these computations the research was carried out using the ASReview software by Utrecht University, which has a simulation mode that you can just input your labelled review file into and perform a simulation study with it. To be found on GitHub. It has many adjustable components/is very versatile.

Then if we still have time left we explore the effect of another feature extraction method, namely doc2vec. This might possibly increase performance as it performs better at grasping structure/hidden relations/hierarchy between words in the texts. So, it could be interesting in more ‘fuzzy’ research areas. A downside is that it takes more computing time.

Then if there’s even more time we might want to compare a different balance strategy.

A SR can be divided into phases. Everything starts with a **systematic search**, leading to then citation screening is performed, then full-text screening (PRISMA-P Group et al. 2015)

What must be the objective of our tool?: It is the tedious task citation screening part where loads of time can be saved.

models are designed in a ‘realistic’ way (you have some inclusions)

Selecting papers is a two-step process: abstract & fulltext screening

Binary classification problem. predict whether a paper in the pool is an inclusion or exclusion, based on labeled instances from and use training data to understand how input variables are related to class.

We’re building active learning model who takes X as an input an predicts class using labels from training set. Model improves by deciding asking more information from the reviewer (oracle), accounting for imbalance.

The researcher has some prior knowledge about the pool, some papers ought to be included in the SR.

#### Assumptions

1. decisions of the original SR are **ground truth** (benchmark) (oracle)

The inclusion rate is … data is imbalanced. what is the philosophy False negatives must be avoided … The cost of a false negative outweighs the cost of a false positive. Note that we assume the oracle/original user to hold the truth. This is of course not always the case.

There are two classes in the data: exlusions and inclusions. The inclusions are clearly the minority class.

Datasets from the medical and social sciences, software engineering and public administration. Medical sciences SRs are viewed as more ‘strict’/‘structured’ and social sciences more messy.

## active learning for systematic reviews

corpus = all the text:

Active learning = increasing classification performance with every query. The query strategy determines the way unlabeled papers are queried to the researcher.

(Danka and Horvath, n.d.)

RQ1 - what are good classifiers RQ2 - what are good optimization strategies

## Background

(O’Mara-Eves et al. 2015) literature review

(Yu, Kraft, and Menzies 2018), (Yu and Menzies 2019) simulated 32 svm classifiers, on software engineering. A popular classifier is SVM. succes with HUTM (fastread), uncertainty, mix of weighting and agressive undersampling, In terms of Yu et al, we adopt .CT.

SVM - tf-idf on medical data, uncertainty sampling, agressive undersampling. (Wallace et al. 2010)

abstrackr

SVM + Weighting + uncertainty (bow) produced good methods (Miwa et al. 2014) Also include social sciences data besides medical data.

(Cohen et al. 2006) perceptron-based classifier (neural network)

SVM on legal documents (no balancing, certainty ) (Cormack and Grossman 2014) in limitations section mentions that LR yields about same results, nb inferior results.

(Kilicoglu et al. 2009) - SVM, naive bayes, boosting and combinations. future work should optimize parameters. “Regarding the base classifiers used in identifying method- ologically rigorous studies, boosting consistently strikes the best balance between precision and recall, whereas naive Bayes in general performs well on recall (demonstrating a tradeoff between recall and precision), as does polynomial SVM on precision. The AUC results are mixed, although boosting has a slight edge overall. These results demonstrate that different classifiers can be used to satisfy different information needs (SVM for specificity, naive Bayes for sensitivity, and boosting for balance between the two, for example).”

Our extensions is that we try different classifiers, on more datasets.

This study was approved by the Ethics Committee of the Faculty of Social and Behavioural Sciences of Utrecht University, filed as an amendement under study 20-104.

All simulations were run using through cartesius EINF-156 Goal: evaluate performance of different models of the ASReview tool. The screening process is simulated using ASReview, seeing if the original inclusions replicate. What would happen if the citation screening would have been performed using asreview? All datasets accompanying the systematic reviews are openly published. We built several machine learning models to perform automated systematic reviews, who are then applied on existing systematic reviews.

When no balancing is applied, the training data set = labeled data set The model can query the labels, who serve as the reviewer, active learning then perform active learning to detect inclusions.

a machine learning-based citation classification tool to reduce workload in systematic reviews of drug class efficacy. Using a perceptron classifier, [WSS@95](mailto:WSS@95)% = 56.61 in (Cohen et al. 2006). (5x2 crossvalidation). Can we beat this? The data

# Methods

Performance of several machine learning models was demonstrated on six systematic review datasets.

## Models

*x* machine learning models () were built. The models consist of multiple components, of which the classifier and the feature extraction strategy varied over the models.

#### Classifiers

Every model applies a classifier to predict the relevance of publications in the data. The classfiier predicts the class of a publication in the dataset expediting a training dataset , predicts the class of an instance given input features

Logistic Regression (LR) - L2-regularized logistic regression. Logistic models the posterior directly, naive bayes has higher bias but lower variance, logistic might perform better when trainign set increases. (Ng and Jordan 2002) Naive Bayes (NB) - Naive Bayes assumes all features are independent given the class value. This is obviously not the case but still the algorithm performs impressively (Zhang 2004). Especially at … tasks.

Random Forests (RF) is where a large number of decision trees are fit on bootstrapped samples of the original data. All trees cast a vote on the class, which are aggregated into a class prediction for each input (Breiman 2001).

Support Vecor Machine (SVM) - finds a multidimensional hyperplane to separate classes. (Tong and Koller 2001)

#### Word representation

To predict the class of a publication class (e.g. whether a publication should be included or excluded), the classifier uses information from the publications in the dataset. Examples of such information are titles and abstracts. However, the classifier cannot predict the publication class from titles and abstracts as they are. Their textual content to needs to be mapped to feature vectors. This process of numerically representing textual content is called ‘word embeddings’.

A classical example of word embeddings a ‘bag of words’ representation. For each text in the data set, the number of occurrences of each word is stored. This leads to n features, where n is the number of distinct words in the texts (Pedregosa et al. 2011). The bag-of-words method is simplistic and will highly value often occuring but otherwise meaningless words such as “and”. Term-frequency Inverse Document Frequency (TF-IDF) (Ramos and others 2003) circumvents this problem by adjusting a term frequency in a text with the inverse docuement frequency, the frequency of a given word in the entire data set.

The model expedits features of previously labeled publications and Based on features of the previously labeled inclusions and exclusions, the model

The features used are title and abstract from every publication. - this is what is prescribed by prisma, (check!)

(TF-IDF) (Doc2Vec)

#### Fixed components

All models apply the same query strategy of certainty sampling, in which the presented publication is always the one of which the model is most certain for it to be relevant.

To decrease the class imbalance in the training data, the model rebalances the training set by Dynamic Supersampling (DS). DS decreases the number of irrelevant papers in the training data, whereas the number of relevant papers are increased (by copy) such that the total number of samples remains the same. The ratio between relevant and irrelevant papers is not fixed, but dynamically updated and depends on the number of training samples, the total number of publications and the ratio between relevant and irrelevant publications.

## Datasets

Six Systematic Reviews from various research areas were used to simulate the automated systematic review process. Data were preprocessed from the original source into a test dataset, containing all publications obtained in the systematic search. The test datasets are labelled indicating the relevant and irrelevant publications. Using these labels, a computer simulation of an automated systematic review can be performed on the systematic review data.

The test datasets contain title information on all publications obtained in the search strategy. The instances in the data consist of a title and an abstract and were labeled to indicate which publications were included in the systematic review. Instances with missing abstracts and duplicate instances were removed from the data. The data preprocessing scripts can be found on the GitHub[[1]](#footnote-36)

Cohen et al. collected systematic review datasets from the medical sciences (Cohen et al. 2006). All systematic reviews in this database are on drug efficacy. The *ace* dataset used in the current study comes from a systematic review on the efficacy of Angiotensin-converting enzyme (ACE) inhibitors.

The *software* dataset is retrieved from (Yu, Kraft, and Menzies 2018), who collected datasets on literature reviews from the software engineering field. This dataset is on fault prediction in software engineering by (Hall et al. 2012).

The *nudging* dataset comes from a review from the behavioural public administration area. The SR includes studies on nudging healthcare professionals (Nagtegaal et al. 2019a). The data was stored on the Harvard Dataverse (Nagtegaal et al. 2019b).

A literature review from field of psychology, *ptsd*. The SR is on studies applying latent trajectory analyses on posttraumatic stress after exposure to trauma (van de Schoot et al. 2017). The corresponding data can be found on the Open Science Framework […].

*wilson*, a dataset from the medical sciences (Appenzeller-Herzog 2020). TA review on effectiveness treatments of Wilson disease (Appenzeller-Herzog et al. 2019).

All datasets started with an initial pool of thousands of papers. A fraction of these papers where deemed relevant for the SR, with inclusion rates around 1-2 percent with one outlier of about 5 percent (Table 1).

Table 1: Statistics on datasets from original systematic reviews.

### Optimizing hyperparameters

Every model component contains hyperparameters, leading to a unique set of hyperparameters for each model. To maximize model performance, we need to find optimal values for the hyperparameters. For every model the optimal hyperparameter values are determined by optimizing on the data . The hyperparameters are optimized by running several hundreds of optimization trials, in which hyperparameter values are sampled from their possible parameter space. A description of all hyperparameters and their sample space can be found in the appendix.

Maximum model performance is defined as the average time it takes to find an inclusion in the data, or more specific: the loss function minimizes the average number of papers needed to screen to find an inclusion (e.g. the area above the curve in the inclusion plot).

The optimization data consists of (a subset from) the six systematic review datasets mentioned above. Three different approaches in composing are explored:

* **one**, where hyperparameters are optimized on only one of the six datasets, . Such hyperparameters are expected to lead to maximum performance in the same dataset .
* **n**, where hyperparameters are optimized on all six data sets, . This optimization approach intends to serve in producing the most optimal hyperparameters overall.
* **n-1**, where hyperparameters are optimized on all six datasets but one, . Serving as a sensitivity analysis for the former condition, e.g. how sensitive are the hyperparamters. also as a cross-validation later on: hyperparameters obtained by training data, test data is never seen before. where d are all datasets but the one where we want to simulate later on. This results in sets of hyperparameters for every model.

Results were visually inspected to check if an optimum (minimal loss) has been reached. More trials were run if the loss still seemed to go down at a quick pace.

The hyperparameter values that were found to lead to a minimum loss value were visually inspected.

### Optimization results

For every model, 13 sets of hyperparameters were optimized.

*to include: plotting the loss reduction over trials*

Optimal values of the hyperparameters were visually inspected.

As an example, the hyperparameters of the RF\_TF-IDF model are presented in Figure 1. A panel displays the optimal values for a certain hyperparameter, where the blue colored dots represent the **one** condition, the green dots the **n-1** condition and the orange dot represents the optimal hyperparameter value when optmizing over all datasets (**n**). The x-axis represents the possible parameter space where the vertical grey lines mark the boundaries of the hyperparameter space (if possible). Note that the feature parameters ngram\_max, split\_ta, use\_keywords and model parameters max\_features and n\_estimators are categorical.

Overall, the optimal values are distributed over the parameter space. Outlying values all belong to the **one** condition where optimization was dependent on one dataset only. Fore example, the class\_weight parameter has an outlying value of 11.3 that belongs to the nudging dataset. (Why this is the case I can only speculate, but it is worth mentioning that this dataset has a relatively high inclusion rate of 5.41%, compared to the other datasets).

![](data:application/eps;base64,)

## Appendix x - Hyperparameters and their sample space

Classifier hyperparameters

class\_weight: normal(0,1) constrained to be > 0  
class weight of the inclusions.

Logistic Regression

c: float. normal(0,1), constrained to be > 0

Support Vector Machine

gamma: ["auto", "scale"],  
Gamma parameter of the SVM model.  
  
C:   
C parameter of the SVM model.  
  
kernel:  
SVM kernel type.["linear", "rbf", "poly", "sigmoid"]

Naive Bayes

"model\_param.alpha", # exp(normal(0,1))

Random Forest

max\_features: int (between 6 and 10)  
 Number of features in the model.  
  
n\_estimators: int between 10 and 100   
 Number of estimators.  
  
model\_param.n\_estimators" #quniform(10,100,1)

Balance strategy hyperparameters: Dynamic supersampling:

a: float  
 Governs the weight of the 1's. Higher values mean linearly more 1's  
 in your training sample.  
alpha: float  
 Governs the scaling the weight of the 1's, as a function of the  
 ratio of ones to zeros. A positive value means that the lower the  
 ratio of zeros to ones, the higher the weight of the ones.  
b: float  
 Governs how strongly we want to sample depending on the total  
 number of samples. A value of 1 means no dependence on the total  
 number of samples, while lower values mean increasingly stronger  
 dependence on the number of samples.

Feature extraction strategy hyperparameters

split\_ta: 0 or 1  
 whether titles and abstracts are split   
   
use\_keywords: 0 or 1  
 whether keywords should be used

TF-IDF

ngram\_max: 1, 2 or 3  
 Can use up to ngrams up to ngram\_max. For example in the case of   
 ngram\_max=2, monograms and bigrams could be used.

Doc2Vec

vector\_size: int (between 32 and 127)  
 Output size of the vector.  
  
epochs: int (between 20 and 50)  
 Number of epochs to train the doc2vec model.  
  
  
min\_count: int (between 1 and 3)  
 Minimum number of occurences for a word in the corpus for it to be   
 included in the model.  
  
window: int (between 5 and 9)  
 Maximum distance over which word vectors influence each other.  
  
dm\_concat: int 0 or 1  
 Whether to concatenate word vectors or not.  
   
dm: int   
 Model to use.  
 0: Use distribute bag of words (DBOW).  
 1: Use distributed memory (DM).  
 2: Use both of the above with half the vector size and concatenate them.  
  
dbow\_words: int 0 or 1  
 Whether to train the word vectors using the skipgram method.

## Simulations

To demonstrate the models, simulations are performed of the models on six original systematic reviews.  
With the labeled datasets, the systematic review can be reproduced but with now using a machine learning model to optimize the reviewing process.

The initial training set of the active learning model in the simulation is one prior included publication and one prior excluded publication, randomly sampled from the data. This demonstrates a ‘worst case scenario’ where the reviewer has minimal prior knowledge on publications in the data. For this simulations there is no need for a human reviewer since the labels in the data can be queried. The oracle is not the reviewer but the labels in the data. The model retrains every time after a label is provided by the oracle. The simulation ends when all publications in the dataset have been queried.

To account for variance, every simulation is repeated for 15 trials .

### Statistical analysis

Results are aggregated over 15 runs of every simulation.

#### measures

#### comparison over conditions

# The software

ASReview takes the following parameters/arguments: We now have 75 combinations. for every for every model (5), for every dataset (5) and for every set of optimized hyperparameters (3), a simulation study consisting trials is performed. From these simulation studies, performance of the different models is evaluated.

Use these inputs to predict relevance of papers.

### Stage 1: hyperparameter optimization

Or, more specific:

#### Performance metrics

For each model, Several metrics are used to compare performance of different models over datasets,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dataset |  | Naive Bayes | Random Forests | Support Vector Machine | Logistic Regression | Dense Neural Network |
| ptsd |  | ? |  |  |  |  |
| ace |  | ? |  |  |  |  |
| hall |  | ? |  |  |  |  |
| nagtegaal |  | ? |  |  |  |  |
| …. |  | ? |  |  |  |  |

The goal is twofold: we want to identify all relevant papers, as fast as we can. Tradeoff: identifying all relevant papers and reducing workload. A good metric to evaluate this is..

What is more important: recall or precision?

Recall more highly valued than precision.

What about class imbalance?

#### RRF

Amount of relevant references found after having screened a certain percentage of the total number of abstracts.

#### Work saved over sampling (WSS)

Indicates how much time can be saved, at a given level of recall. WSS is in terms of the percentage of abstracts that don’t have to be screened by the researcher. Typically, WSS is measured at a recall of 0.95. Reasonable because..

#### Raoul

#### Utility?

#### F-measure

#### ROC/AUC

Is performance related to some characteristic (n, inclusion rate, …)

? How to compare outcomes of 3 different optimization strategies?

# Results

ace

nudging

ptsd

software

virus

wilson

BCTD

0

0

0

0

0

0

RCTD

0

0

0

0

0

0

SCTD

0

0

0

0

0

0

LCTD

0

0

0

0

0

0

NCTD

0

0

0

0

0

0

BCDD

0

0

0

0

0

0

RCDD

0

0

0

0

0

0

SCDD

0

0

0

0

0

0

LCDD

0

0

0

0

0

0

NCDD

0

0

0

0

0

0

# Discussion

# Appendix A - list of definitions

### Feature Extraction Strategies

split\_ta = overall hyperparameter

#### TF-IDF

##### hyperparameters

ngram\_max: int Can use up to ngrams up to ngram\_max. For example in the case of ngram\_max=2, monograms and bigrams could be used.

#### Doc2Vec

Predicts words from context. Aims at capturing the relations between word (man-woman, king-queen). (Le and Mikolov 2014). Using a neural network.

using Continuous Bag-of-Words (CBOW), Skip-Gram model, …. Word vector *W* and extra: document vector *D*, trained to predict words in the text.

From gensim (Řehůřek and Sojka 2010).

Arguments  
 ---------  
 vector\_size: int  
 Output size of the vector.  
 epochs: int  
 Number of epochs to train the doc2vec model.  
 min\_count: int  
 Minimum number of occurences for a word in the corpus for it to  
 be included in the model.  
 workers: int  
 Number of threads to train the model with.  
 window: int  
 Maximum distance over which word vectors influence each other.  
 dm\_concat: int  
 Whether to concatenate word vectors or not.  
 See paper for more detail.  
 dm: int  
 Model to use.  
 0: Use distribute bag of words (DBOW).  
 1: Use distributed memory (DM).  
 2: Use both of the above with half the vector size and concatenate  
 them.  
 dbow\_words: int  
 Whether to train the word vectors using the skipgram metho

#### SBERT

BERT-base model with mean-tokens pooling (Reimers and Gurevych 2019)

#### embeddingIdf

This model averages the weighted word vectors of all the words in the text, in order to get a single feature vector for each text. The weights are provided by the inverse document frequencies

### Models

#### Naive Bayes

Naive Bayes assumes all features are independent given the class value. (Zhang 2004)

ASReview uses the MultinomialNB from the scikit-learn package (Pedregosa et al. 2011), that implements the naive Bayes algorithm for multinomially distributed data. nb

Hyperparameters

* alpha - accounts for features not present in learning samples and prevents zero probabilities in further computations.

#### Random Forests

A number of decision trees are fit on bootstrapped samples of the original data, (Breiman 2001) RandomForestClassifier from sklearn

Arguments ——— n\_estimators: int Number of estimators. max\_features: int Number of features in the model. class\_weight: float Class weight of the inclusions. random\_state: int, RandomState Set the random state of the RNG. """

#### Support Vector Machine

Arguments ——— gamma: str Gamma parameter of the SVM model. class\_weight: class\_weight of the inclusions. C: C parameter of the SVM model. kernel: SVM kernel type. random\_state: State of the RNG.

#### Logistic Regression

#### **Dense Neural Network**

### Query Strategies

* Max - Choose the most likely samples to be included according to the model
* Uncertainty - choose the most uncertain samples according to the model (i.e. closest to 0.5 probability) (Lewis and Catlett 1994)
* Random - randomly selects abstracts with no regard to model assigned probabilities.
* Cluster - Use clustering after feature extraction on the dataset. Then the highest probabilities within random clusters are sampled

The following combinations are simulated:

* cluster
* max
* cluster \* random
* cluster \* uncertainty
* max \* cluster
* max \* random
* max \* uncertainty

### Balance Strategies

### amount of training data

* n\_instances = number of papers queried each query
* n\_queries = number of queries
* n\_prior\_included: 5
* n\_prior\_excluded:

# Combinations

This leads to 119 combinations of configurations.

* Naive bayes only goes with tfidf feature extraction.
* For the feature extraction strategies we will focus on doc2vec and tfidf. (but will compute all 4)
* This leads to 3 \* 7 \* 4 \* 3 + 1 \* 7 \* 1 \* 3 = 273 combinations.

See appendix A for a table containing all 273 combinations.

## Cross-validation

Should give an accurate estimate of maximum performance / future systematic reviews to be performed.

# Appendix B - combinations

# Appendix C - supercomputer Cartesius

500,000 SBU

Running on Cartesius is charged in System Billing Units (SBUs), and charging is based on the wall clock time of a job. On fat and thin nodes, an SBU is equal to using 1 core for 1 hour (a core hour), or 1 core for 20 minutes on a GPU node. Since compute nodes are allocated exclusively to a single job at a time, you will be charged for all cores on that node - even if you are using less.

In the current study, the classifier and the feature extraction strategy are varied, whereas the query and balance strategy remain fixed. In the current study only a fraction of all possible configurations are tested for the sake of brevity. There are many more options available and open to exploration.

# References

Appenzeller-Herzog, Christian. 2020. “Data from Comparative Effectiveness of Common Therapies for Wilson Disease: A Systematic Review and Meta-Analysis of Controlled Studies.” Zenodo.

Appenzeller-Herzog, Christian, Tim Mathes, Marlies L. S. Heeres, Karl Heinz Weiss, Roderick H. J. Houwen, and Hannah Ewald. 2019. “Comparative Effectiveness of Common Therapies for Wilson Disease: A Systematic Review and Meta-Analysis of Controlled Studies.” *Liver International* 39 (11): 2136–52. <https://doi.org/10.1111/liv.14179>.

Breiman, Leo. 2001. “Random Forests.” *Machine Learning* 45 (1): 5–32. <https://doi.org/10.1023/A:1010933404324>.

Cohen, A. M., W. R. Hersh, K. Peterson, and Po-Yin Yen. 2006. “Reducing Workload in Systematic Review Preparation Using Automated Citation Classification.” *Journal of the American Medical Informatics Association : JAMIA* 13 (2): 206–19. <https://doi.org/10.1197/jamia.M1929>.

Cormack, Gordon V., and Maura R. Grossman. 2014. “Evaluation of Machine-Learning Protocols for Technology-Assisted Review in Electronic Discovery.” In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 153–62. SIGIR ’14. Gold Coast, Queensland, Australia: Association for Computing Machinery. <https://doi.org/10.1145/2600428.2609601>.

Danka, Tivadar, and Peter Horvath. n.d. “modAL: A Modular Active Learning Framework for Python.”

Hall, Tracy, Sarah Beecham, David Bowes, David Gray, and Steve Counsell. 2012. “A Systematic Literature Review on Fault Prediction Performance in Software Engineering.” *IEEE Transactions on Software Engineering* 38 (6): 1276–1304. <https://doi.org/10.1109/TSE.2011.103>.

Kilicoglu, H., D. Demner-Fushman, T. C. Rindflesch, N. L. Wilczynski, and R. B. Haynes. 2009. “Towards Automatic Recognition of Scientifically Rigorous Clinical Research Evidence.” *Journal of the American Medical Informatics Association* 16 (1): 25–31. <https://doi.org/10/bjkhh9>.

Le, Quoc V., and Tomas Mikolov. 2014. “Distributed Representations of Sentences and Documents.” *arXiv:1405.4053 [Cs]*, May. <http://arxiv.org/abs/1405.4053>.

Lewis, David D., and Jason Catlett. 1994. “Heterogeneous Uncertainty Sampling for Supervised Learning.” In *Machine Learning Proceedings 1994*, edited by William W. Cohen and Haym Hirsh, 148–56. San Francisco (CA): Morgan Kaufmann. <https://doi.org/10.1016/B978-1-55860-335-6.50026-X>.

Miwa, Makoto, James Thomas, Alison O’Mara-Eves, and Sophia Ananiadou. 2014. “Reducing Systematic Review Workload Through Certainty-Based Screening.” *Journal of Biomedical Informatics* 51 (October): 242–53. <https://doi.org/10.1016/j.jbi.2014.06.005>.

Nagtegaal, Rosanna, Lars Tummers, Mirko Noordegraaf, and Victor Bekkers. 2019a. “Nudging Healthcare Professionals Towards Evidence-Based Medicine: A Systematic Scoping Review.” *Journal of Behavioral Public Administration* 2 (2). <https://doi.org/doi.org/10.30636/jbpa.22.71>.

———. 2019b. “Nudging Healthcare Professionals Towards Evidence-Based Medicine: A Systematic Scoping Review.” Harvard Dataverse.

Ng, Andrew Y., and Michael I. Jordan. 2002. “On Discriminative Vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes.” In *Advances in Neural Information Processing Systems 14*, edited by T. G. Dietterich, S. Becker, and Z. Ghahramani, 841–48. MIT Press.

O’Mara-Eves, Alison, James Thomas, John McNaught, Makoto Miwa, and Sophia Ananiadou. 2015. “Using Text Mining for Study Identification in Systematic Reviews: A Systematic Review of Current Approaches.” *Systematic Reviews* 4 (1): 5. <https://doi.org/10.1186/2046-4053-4-5>.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.

PRISMA-P Group, David Moher, Larissa Shamseer, Mike Clarke, Davina Ghersi, Alessandro Liberati, Mark Petticrew, Paul Shekelle, and Lesley A Stewart. 2015. “Preferred Reporting Items for Systematic Review and Meta-Analysis Protocols (PRISMA-P) 2015 Statement.” *Systematic Reviews* 4 (1): 1. <https://doi.org/10.1186/2046-4053-4-1>.

Ramos, Juan, and others. 2003. “Using Tf-Idf to Determine Word Relevance in Document Queries.” In *Proceedings of the First Instructional Conference on Machine Learning*, 242:133–42. Piscataway, NJ.

Reimers, Nils, and Iryna Gurevych. 2019. “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks.” *arXiv:1908.10084 [Cs]*, August. <http://arxiv.org/abs/1908.10084>.

Řehůřek, Radim, and Petr Sojka. 2010. “Software Framework for Topic Modelling with Large Corpora.” In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50. Valletta, Malta: ELRA.

Tong, Simon, and Daphne Koller. 2001. “Support Vector Machine Active Learning with Applications to Text Classification.” *Journal of Machine Learning Research* 2 (Nov): 45–66.

van de Schoot, Rens, Marit Sijbrandij, Sonja D. Winter, Sarah Depaoli, and Jeroen K. Vermunt. 2017. “The GRoLTS-Checklist: Guidelines for Reporting on Latent Trajectory Studies.” *Structural Equation Modeling: A Multidisciplinary Journal* 24 (3): 451–67. <https://doi.org/10/gdpcw9>.

Wallace, Byron C., Thomas A. Trikalinos, Joseph Lau, Carla Brodley, and Christopher H. Schmid. 2010. “Semi-Automated Screening of Biomedical Citations for Systematic Reviews.” *BMC Bioinformatics* 11 (1): 55. <https://doi.org/10.1186/1471-2105-11-55>.

Yu, Zhe, Nicholas A. Kraft, and Tim Menzies. 2018. “Finding Better Active Learners for Faster Literature Reviews.” *Empirical Software Engineering* 23 (6): 3161–86. <https://doi.org/10.1007/s10664-017-9587-0>.

Yu, Zhe, and Tim Menzies. 2019. “FAST2: An Intelligent Assistant for Finding Relevant Papers.” *Expert Systems with Applications* 120 (April): 57–71. <https://doi.org/10.1016/j.eswa.2018.11.021>.

Zhang, Harry. 2004. “The Optimality of Naive Bayes.” In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*. Vol. 2.

1. <https://github.com/GerbrichFerdinands/asreview-thesis> [↑](#footnote-ref-36)