Int. J. Appl. Math. Comput. Sci., 2023, Vol. 33, No. 4, 603–621
DOI: 10.34768/amcs-2023-0043

BAG OF WORDS AND EMBEDDING TEXT REPRESENTATION METHODS FOR MEDICAL ARTICLE CLASSIFICATION

PAWEŁ CICHOSZ [a]

[a]Institute of Computer Science
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
e-mail: pawel.cichosz@pw.edu.pl

Text classification has become a standard component of automated systematic literature review (SLR) solutions, where articles are classified as relevant or irrelevant to a particular literature study topic. Conventional machine learning algorithms for tabular data which can learn quickly from not necessarily large and usually imbalanced data with low computational demands are well suited to this application, but they require that the text data be transformed to a vector representation. This work investigates the utility of different types of text representations for this purpose. Experiments are presented using the bag of words representation and selected representations based on word or text embeddings: *word2vec*, *doc2vec*, *GloVe*, *fastText*, *Flair*, and *BioBERT*. Four classification algorithms are used with these representations: a naive Bayes classifier, logistic regression, support vector machines, and random forest. They are applied to datasets consisting of scientific article abstracts from systematic literature review studies in the medical domain and compared with the pre-trained *BioBERT* model fine-tuned for classification. The obtained results confirm that the choice of text representation is essential for successful text classification. It turns out that, while the standard bag of words representation is hard to beat, *fastText* word embeddings make it possible to achieve roughly the same level of classification quality with the added benefit of much lower dimensionality and capability of handling out-of-vocabulary words. More refined embeddings methods based on deep neural networks, while much more demanding computationally, do not appear to offer substantial advantages for the classification task. The fine-tuned *BioBERT* classification model performs on par with conventional algorithms when they are coupled with their best text representation methods.

Keywords: text representation, text classification, bag of words, word embeddings.

1. Introduction

The rapidly growing amount of available textual data as well as the progress in machine learning and natural language processing make text classification an interesting research direction and an area of successful practical applications (e.g., McCallum and Nigam, 1998; Joachims, 1998; Radovanović and Ivanović, 2008; Dařena and Žižka, 2017; Zymkowski *et al.*, 2022). These research studies and applications deal with such diverse types of text as email messages, online chats, website contents, blog posts, forum discussions, customer opinions, product reviews, or scientific articles. For the latter, text classification has become a standard component of automated systematic literature review (SLR) solutions, where articles are classified as relevant or irrelevant to a particular literature study topic (García Adeva *et al.*, 2014; van den Bulk *et al.*, 2022).

1.1. Motivation. This work focuses on the medical domain because it is a part of a bigger research project on automating systematic literature reviews for food safety studies. However, medical article classification deserves special interest for other reasons as well. It is one of those application domains where large-scale literature screening is routinely performed not only by individual researchers or academic research teams but also by public and private health care institutions, and the quality of classification is of utmost importance, since it may have consequences to human health. Medical articles may be also quite challenging to classify, with subtle, hard to capture class boundaries. Examining the utility of text representation methods for this domain may therefore provide more useful and interesting findings than for some other more popular but not so demanding applications, such as product review or social media post sentiment classification or news article topic classification. Last but not least, the availability of the domain-specific pre-trained *BioBERT* deep-learning language model (Lee *et al.*, 2020), derived from *BERT* (Devlin *et al.*, 2019), which can be used to provide one possible type of text representation, but also (after fine-tuning) serve as an alternative classification algorithm, makes studying this domain even better justified.

Scientific article classification with respect to relevance may appear easier than the classification of other types of text, as it may be less prone to being misled by such issues as subjectivity, word sense ambiguity, irony, etc., and less affected by imperfections such as spelling and grammar errors, sloppy and concise style, overusing of slang, colloquial and idiomatic expressions etc. It is also less likely, although still possible, that the relevance depends on specific word order or context within sentences. It is therefore not essential to discover and use hidden semantic information that may be hard to capture, particularly for models that use word occurrence statistics for text classification. On the other hand, there are some inherent challenges related to scientific article classification that may be not so common for other types of text, such as a relatively small number of labeled training instances (because, to be labeled, an article needs to be screened by a human expert to determine relevance to a particular topic), poor separation between classes (because relevant and irrelevant articles are usually from the same thematic area and may be quite similar), heavily imbalanced classes (because the number of irrelevant articles often vastly exceeds that of relevant ones), and the requirement to create models quickly using limited computational resources (because model creation takes place as part of an interactive session with the user).

The non-necessity of unraveling deep semantics and the necessity to learn quickly from not necessarily large and usually imbalanced data with low computational demands may favor conventional machine learning algorithms for tabular data, rather than deep learning neural networks (Borisov *et al.*, 2022). While the latter are increasingly popular and often highly successful in complex natural language processing tasks, the former still remain valid contenders for the classification task, using a fraction of computational resources. Active learning techniques that may reduce the number of labeled training instances and the associated human expert effort required to achieve high quality models are also mostly developed for and easier to apply with conventional learning algorithms (Ren *et al.*, 2020). Since active learning involves many iterations of model creation and prediction, interleaved with user interaction, their computational efficiency and capability to run on low-cost CPU-only hardware becomes even more important. These algorithms, however, require that the text data for both model creation and prediction be transformed to a vector representation. The quality of models that they can produce may heavily depend on the type and dimensionality of such a representation, as well as the specific transformation method.

1.2. Related work. Text classification has been one the the most extensively studied text mining tasks (Yang and Pedersen, 1997; McCallum and Nigam, 1998; Joachims, 1998; Forman, 2003; Hassan *et al.*, 2007; Radovanović and Ivanović, 2008; Aggarwal and Zhai, 2012; Dařena and Žižka, 2017; Zymkowski *et al.*, 2022). However, not so many of those prior studies focused on the topic of the utility of different text representation methods in combination with conventional classification algorithms for tabular data. The most closely related publications are presented in Table 1, highlighting text representation methods and classification used as well as metric employed for predictive performance evaluation and data on which the experiments are performed.

These related studies bring interesting and useful results which encourage further exploration of the usefulness of embedding-based representations for text classification, but each of them remains limited in one way or another. They only use either pre-trained or custom embedding vectors or models, without comparing these two. Some of them do not compare embedding-based representations to bag of words, which is known to work well and often remains the most obvious choice. Some of them do not use conventional (non-neural) learning algorithms with embedding-based representations. This is a particularly important omission, since the utility of embeddings for conventional learning algorithms is arguably more interesting and worthwhile to investigate than for deep neural networks, which develop internal data representations anyway. While some of these prior studies evaluate the predictive performance using the area under the ROC curve, which aggregates over all possible model operating points, most of them focus on less informative single-point metrics such as accuracy or F1, and none of them uses the area under the precision-recall curve which is more sensitive to false positives with imbalanced data and therefore likely to reveal more performance differences between text representation methods and classification algorithms. Last but not least, these related studies are either limited to a single dataset from a particular application domain or to a few benchmark datasets, which is hardly sufficient to draw general conclusions.

1.3. Contributions. This work attempts to fulfill the gap left by prior work by presenting systematic

Table 1. Related work.

Work	Text representations	Classification algorithms	Evaluation	Data
Cichosz, 2018	bag of words, custom *GloVe*	naive Bayes, logistic regression, SVM, random forest	ROC AUC	discussion forum posts
Kaibi *et al.*, 2019	custom *word2vec*, *fastText*, *GloVe*	naive Bayes, SVM, logistic regression,random forest	precision, recall, F1	Twitter sentiment
Helaskar and Sonawane, 2019	bag of words, custom *word2vec*	SVM	accuracy, precision, recall	Reuters
Wang *et al.*, 2020	pre-trained *word2vec*, *GloVe*, *fastText*, *char*, *ELMo*, *BERT*	deep neural networks (CNN, Bi-LSTM)	accuracy, F1	20-Newsgroups, SST-2, AAPD, Reuters
Deb and Chanda, 2022	bag of words, pre-trained *word2vec*, *GloVe*, *fastText*, *BERT*	for bag of words: decision trees, random forest, logistic regression; for embeddings: deep neural networks (softmax, Bi-LSTM)	accuracy, F1, ROC AUC	Twitter disaster

experiments based on the design decisions listed below, which may be considered its distinguishing features,

- Different variants of the bag of words representation are used: term frequency, term frequency-inverse document frequency, with and without lemmatization.
- Different variants of embedding-based representations are used: *word2vec*, *doc2vec*, *fastText*, *Flair*, *BioBERT*, obtained using both publicly available pre-trained embedding vectors or models and custom embedding vectors or models trained on the data to be classified.
- Several classification algorithms are used that are known to work well with text data: naive Bayes classifier, logistic regression, SVM, and random forest.
- Conventional classification algorithms coupled with the best performing text representations are compared with the pre-trained *BioBERT* model fine-tuned for classification.
- Experiments are performed on 15 datasets from medical systematic literature review studies, with class labels representing relevance or irrelevance to the study topic. The datasets are of of varying size (between about 300 and 3000) and class imbalance level (with the relevant class percentage between about 2% and 35%).
- Predictive performance is evaluated with respect to the area under ROC and precision-recall curves using 5×10-fold cross-validation, with a bootstrap significance test for text representation and classification algorithm comparisons.
- The experimental procedure is described in detail, including software libraries and algorithm configurations.

These design decisions make it possible to obtain the following contributions:

- A systematic evaluation of the predictive performance of bag of words and several types of word or text embeddings with different conventional classification algorithms applied to medical article classification.
- Rankings of text representation methods and classification algorithms with respect to predictive performance.
- An identification of the best performing text representation methods for particular classification algorithms and the best performing classification algorithms for particular text representation methods.
- A comparison with a domain-specific pre-trained deep learning biomedical language model fine-tuned for the classification task.

This improves the state of knowledge in text classification methods and may provide practically useful guidelines for their applications.

The remainder of this article is organized as follows. Section 2 presents the text representation methods used in this work. Algorithms applied to create classification models are briefly described in Section 3. Section 4 provides the details of the experimental procedure and presents the obtained results. Section 5 summarizes the major findings as well as limitations of this work and outlines possible continuation directions.

2. Text representation methods

The first issue to be resolved when applying modeling algorithms from the fields of machine learning and statistics to text data is transforming the analyzed document corpus into a representation that can be handled by these algorithms. This is usually a vector representation in which each document is assigned values of a fixed, common set of attributes (Dumais *et al.*, 1998; Szymański, 2014). A document is then represented by a vector of its attribute values, with the number of vector elements being the same for all documents. This work uses the simple and highly popular bag of words representation and several types of word or text embeddings, i.e., transformations that map single words or sequences of words into a multidimensional real-valued vector space of fixed dimensionality. The latter include approaches using shallow and deep neural models (Babić *et al.*, 2020).

2.1. Bag of words. The simplest vector text representation that remains popular in text classification applications is the *bag of words* (BOW) representation (McCallum and Nigam, 1998; Joachims, 1998; Aggarwal and Zhai, 2012), in which attributes directly correspond to words or, in a slightly more general setting, n-*grams*—word sequences of length n (usually $n \leq 3$). Words or n-grams used for this representation are called *terms*. In the most common term frequency (TF) variant, for each term the value of the corresponding attribute is the number of the term's occurrences in a given document. A TF-IDF variant is also popular in which term frequencies are normalized by inverse document frequencies to downweight terms occurring in many documents and upweight terms unique to smaller subsets of documents (Salton and Buckley, 1988). The dimensionality of the representation can be controlled by frequency-based term filtering. Lemmatization can be applied to assign a single term to different inflected forms of a word.

2.2. *Word2vec* and *doc2vec*. The *word2vec* representation is the oldest but still popular type of word embeddings in which a mapping of words to vectors is obtained using a shallow two-layer neural network (Mikolov *et al.*, 2013). The network is trained by processing word sequences occurring in fixed-length context windows in one of the following two possible modes:

***continuous bag of words* (CBOW):** predicting the current input word based on the words occurring in the corresponding context window,

***skip-gram* (SG):** predicting the words occurring in the context window based on the current input word.

Of those, the former is faster and recommended for bigger datasets, but the latter may more adequately represent the context of less frequent words. In any case, the network is trained on a text corpus and then used to extract word vectors.

Word vectors can be used to obtain document vectors for text classification by averaging: the document vector is obtained as the weighted average of word vectors for all words occurring in the document and present in the vocabulary, with their occurrence counts serving as weights (Mitchell and Lapata, 2010; Yessenalina and Cardie, 2011). It makes sense to L2-normalize word vectors (i.e., scale them to a unit length) prior to this averaging so that the impact of individual words on the document vector is only proportional to their occurrence counts.

A more refined and potentially more useful approach is a modified variation of the algorithm referred to as *doc2vec* (Le and Mikolov, 2014). As original *word2vec* predicts word occurrences within contexts of surrounding words, thus identifying word vectors, the *doc2vec* algorithm predicts word occurrences based on both the context of surrounding words and of the document, thus identifying both word vectors and document vectors.

2.3. *GloVe*. Global vectors or *GloVe* word embeddings are determined based on term co-occurrence statistics (Pennington *et al.*, 2014). The algorithm creates a co-occurrence matrix, containing, for all term pairs, the number of occurrences of one term within a context window of the other term. The matrix can be used to estimate co-occurrence probabilities, i.e., for all term pairs, the probability of one term appearing in the context of the other term. The *GloVe* algorithm operates by factorizing the co-occurrence matrix using stochastic gradient descent to determine term vectors such that the dot product of vectors for any two terms approximates their co-occurrence probability.

Like *word2vec* (and unlike *doc2vec*), the *GloVe* representation produces word vectors. Document vectors can be obtained by averaging, as described above for *word2vec*.

2.4. *FastText*. The *fastText* algorithm creates word embeddings in a similar way as *word2vec*, with one major difference that may lead to important practical advantages. To derive word representations it goes down to the character level, treating each word as a set of character n-grams (typically for n between 3 and 6), corresponding to subwords. It then applies either the CBOW or SG training mode to determine embedding vectors for such character n-grams and additionally to full words, and combines them into word vectors by summation (Bojanowski *et al.*, 2016). The CBOW mode, similarly to *word2vec*, predicts a target word based on words occurring in its context window. The SG mode predicts a target word based on another nearby word, which is similar to the co-occurrence approach of *GloVe* and found to work better with subwords. An embedding vector for a sequence of multiple words (a sentence, a paragraph, or a document) is obtained by averaging L2-normalized vectors for each of these words.

Morphologically similar words have some shared character n-grams and therefore their resulting *fastText* representations are partially shared. This makes a more effective use of training data and permits learning more adequate representations of even rarely occurring words. Useful word embeddings can be obtained using relatively small datasets, which is useful for creating custom domain-specific text representation. Word vectors can be obtained even for previously unseen words, which is very convenient when applying a classification model using text representation derived from available data to a new dataset.

2.4.1. *Flair*.

The type of word embeddings referred to as *Flair* due to the name of the natural language processing software library that provides its implementation (Akbik *et al.*, 2019) can be described as contextual string embeddings. It takes the character-level approach of *fastText* even further, by abandoning any explicit notion of words which are only considered sequences of characters, but combines it with contextualization by surrounding text (Akbik *et al.*, 2018). Such embeddings are determined based on hidden states of a recurrent neural network character-level language model, using the LSTM architecture (Graves, 2013). It can be a forward model that predicts the probability of the next character based on the sequence of previous characters, or a backward model that works in the reverse direction. Word embeddings are obtained by concatenating network hidden states. A combination of a forward model and a backward model can be used by the so called stacking, which basically concatenates embedding vectors generated using the two models.

Flair embeddings can be obtained for arbitrary sequences of characters, including rarely occurring or previously unseen words. However, each word is processed as part of a longer sequence (a sentence or a paragraph) and different word vectors are obtained for the same words in different contexts. This makes contextual string embeddings potentially more powerful than more basic word embeddings reviewed above. However, they are also substantially more costly to train and to calculate—the former requires training a recurrent neural network with hundreds or thousands of hidden units, and the latter—processing text through such a network character by character.

2.5. *BioBERT*.

BioBERT is a deep learning language model obtained adopting the architecture and learning algorithm of the *BERT* model (*Bidirectional Encoder Representations from Transformers* (Devlin *et al.*, 2019)) and pre-training it on large biomedical text corpora (Lee *et al.*, 2020). The *BERT* and *BioBERT* models use the transformer network architecture based on the attention mechanism, without recurrence, which makes them faster and easier to train than LSTM networks (Vaswani *et al.*, 2017). Rather than combining a forward model and a backward model, they use a masking approach to obtain a contextual bidirectional model: randomly selected words in a sequence are masked (replaced by a special mask token or by random words) and have to be predicted based on the remaining words. Therefore both the left and right contexts of words are taken into account in all network layers.

BioBERT embeddings for individual words with no context (i.e., single-word sequences) could be aggregated by weighted averaging to obtain document vectors, similarly as for *GloVe* or *word2vec*, but this would loose the context-sensitivity which is supposed to be their substantial advantage over those simpler approaches. For relatively short texts it makes therefore more sense to treat the entire text as one sequence for which the model generates an embedding vector.

3. Classification algorithms

A selection of well known classification algorithms for tabular data that have proved useful for text classification is used for this study a. These algorithms are relatively resistant to overfitting, which is a serious threat in high-dimensional input spaces, and do not require extensive hyperparameter tuning to achieve a reasonable level of prediction quality.

3.1. Naive Bayes.

One of the simplest practically useful classification algorithms, the *naive Bayes classifier*, is often successfully applied to text classification. It predicts the class probability given attribute values using the Bayes theorem and assuming the conditional independence of attribute values given the class. The algorithms does not tend to overfit and requires practically no tuning.

For the bag of words representation, where attributes correspond to word occurrence frequencies, conditional attribute value probabilities given the class can be estimated using the multinomial distribution (McCallum and Nigam, 1998; Lewis, 1998). The resulting version of the algorithm is referred to as the *multinomial* naive Bayes classifier. For representations based on word or text embeddings, where attribute values cannot be interpreted as frequencies, the standard approach of handling numeric attributes using normal density function values can be applied. This is referred to as the *Gaussian* naive Bayes classifier.

3.2. Logistic regression. A logistic regression model combines an inner linear representation function with an outer logit transformation which makes it possible to interpret model outputs as predicted class probabilities (Hilbe, 2009). Training such a model consists in finding parameters of the inner linear function which maximize the log-likelihood of training set classes.

The algorithm is easy to apply and not overly prone to overfitting unless used for high-dimensional data, which is unfortunately often the case in text classification applications, at least with the bag of words representation. Logistic regression can be used with text data in arbitrary vector representations.

3.3. Support vector machines. Support vector machines (SVMs) use a simple linear-threshold model representation in which a linear decision boundary is identified using the objective of classification margin maximization subject to class separation constraints (Cortes and Vapnik, 1995; Platt, 1998; Hamel, 2009). Nonlinear relationships can be represented by input transformation with kernel functions. Signed distances of classified instances from the decision boundary serve as numeric decision function values.

An important advantage of SVMs, achieved by margin maximization, is the insensitivity of model quality to data dimensionality, which—unlike for many other algorithms—does not increase the risk of overfitting. This makes the algorithm well suited to text classification, where high dimensionality is to be expected, at least with the bag of words representation (Joachims, 1998; Joachims, 2002).

3.4. Random forest. The random forest algorithm creates an ensemble model consisting of unpruned decision trees, grown based on multiple bootstrap samples drawn with replacement from the training set, with randomized split selection (Breiman, 2001). Random forest prediction is performed by simple unweighted voting of individual trees, and vote distribution can be used to obtain class probability predictions. With several dozens or hundreds of diversified trees this usually delivers predictions of very high quality.

The algorithm is easy to use due to its resistance to overfitting and rather low sensitivity to hyperparameter settings. It is applicable to text classification with an arbitrary vector representation, which is confirmed by successful results reported in the literature (Rios and Zha, 2004; Koprinska *et al.*, 2007; Xue and Li, 2015).

4. Experimental study

The experimental study applies text representation methods reviewed in Section 2 with classification algorithms briefly described in Section 3 to datasets containing scientific articles from biomedical SLR studies. It also includes a comparison with a pre-trained *BioBERT* model fine-tuned for classification.

4.1. Datasets. The datasets used for the experiments come from drug class review SLR studies (Cohen *et al.*, 2006).[1] There are 15 topics or study areas and articles (identified by PubMed ID) are labeled as relevant or irrelevant to those topics based on human evaluation of abstracts and full texts. Abstracts for most of these articles (with some not matching correctly ignored) have been obtained from the MEDLINE database.[2] Combining article abstracts with relevant/irrelevant class labels for 15 topics yields therefore 15 datasets for text classification.[3] Since full article texts are not available and classification can be performed for abstracts only, relevant/irrelevant class labels based on human evaluation of abstracts are used.

Table 2 presents the size and class distribution of each dataset. They differ slightly from those reported by Cohen *et al.* (2006) due to possible changes of the MEDLINE records or differences in the matching procedure. As to be expected in the case of systematic literature review studies, datasets are imbalanced (some of them quite heavily) and relatively small.

Besides the work of Cohen *et al.* (2006) from which the data origin, they were used in other studies (Matwin *et al.*, 2010; Jonnalagadda and Petitti, 2013; Khabsa *et al.*, 2016; Ji *et al.*, 2017). They focused on workload reduction in a systematic literature review process rather than on predictive performance per se and their results cannot be directly compared with those reported in this article due to different performance measures and model evaluation procedures.

[1]Available from `https://dmice.ohsu.edu/cohenaa/systematic-drug-class-review-data.html` (this and all other mentioned websites last accessed on 30 November 2023).

[2]Strictly speaking, from MEDLINE-exported files for the TREC 2004 Genomics Track, available from `https://dmice.ohsu.edu/trec-gen/2004data.html`.

[3]The datasets prepared and used for this work, containing article PubMed IDs, abstracts, and class labels, are available from `https://doi.org/10.6084/m9.figshare.23626656.v1`.

Table 2. Dataset sizes and class distribution.

Dataset	Size	Relevant %
ACEInhibitors	2215	7.54
ADHD	797	10.41
Antihistamines	285	31.23
AtypicalAntipsychotics	1020	32.45
BetaBlockers	1849	14.49
CalciumChannelBlockers	1100	23.18
Estrogens	344	22.67
NSAIDS	355	23.38
Opiods	1760	2.44
OralHypoglycemics	466	28.11
ProtonPumpInhibitors	1193	18.69
SkeletalMuscleRelaxants	1341	2.24
Statins	2727	5.50
Triptans	585	34.36
UrinaryIncontinence	283	24.03

4.2. Algorithm implementations and setup. The experimental study was, with one minor exception related to *GloVe* embedding vectors training explained later, performed in the Python language environment using Python software libraries.

4.2.1. Text representation methods. Implementation and setup details for all types of text representation described in Section 2 are provided below. Notice that for embedding-based representations the dimensionality depends on available pre-trained embedding vectors or models and is therefore not uniform across different representation types. When using custom embedding vectors or models trained on the data used for these experiments, a uniform dimensionality of 100 is arbitrarily adopted. It is much smaller than for pre-trained embeddings, consistently with the relatively small size of the available training data, and increasing the dimensionality was not found to improve predictive performance in preliminary experiments.

Bag of words. The bag of words representation is implemented using the `CountVectorizer` and `TfidfVectorizer` classes from the `scikit-learn` Python library[4] (Pedregosa *et al.*, 2011). The `min_df` and `max_df` parameters, controlling the minimum and maximum document frequency for a word to be included in the vocabulary, were arbitrarily set to common-sense values 0.01 and 0.95, respectively, resulting in a moderate vocabulary size (typically between 1000 and 2000 words). Prior to determining the vocabulary and counting word occurrences, text was tokenized with non-word and stop-word removal as well as with optional lemmatization using the `spaCy` Python library[5] (Honnibal *et al.*, 2021). The following versions of the bag of words representation are used:

BOW-TF: with TF attribute values,

BOW-TFIDF: with TF-IDF attribute values,

BOW-L-TF: with lemmatization and TF attribute values,

BOW-L-TFIDF: with lemmatization and TF-IDF attribute values.

Word2vec. The 300-dimensional *word2vec* vectors pre-trained on the Google News corpus were used.[6] For training custom vectors the `Word2Vec` class from the `Gensim` Python library was used (Řehůřek and Sojka, 2010; Řehůřek, 2021), with vector dimensionality set to 100, CBOW training mode, and other parameters left at their default settings. The following versions of the representation are used:

W2V-P: pre-trained *word2vec* vectors,

W2V-CS: custom *word2vec* vectors trained on data from the single SLR study for which a classification model is to be created,

W2V-CA: custom *word2vec* vectors trained on combined data from all SLR studies.

Doc2vec. A pre-trained *doc2vec* embedding model appears not to be available. For training custom *doc2vec* embedding models the `Doc2Vec` class from the `Gensim` Python library was used (Řehůřek and Sojka, 2010; Řehůřek, 2021), with vector dimensionality set to 100 and other parameters left at their default settings. The following versions of the representation are used:

D2V-CS: a custom *doc2vec* embedding model trained on data from the single SLR study for which a classification model is to be created,

D2V-CA: a custom *doc2vec* embedding model trained on combined data from all SLR studies.

GloVe. The 300-dimensional *GloVe* vectors pre-trained on the Wikipedia 2014 and Gigaworld 5 corpora were used.[7] For training custom vectors the GloVe C code was used,[8] with vector dimensionality set to 100, context

[4]Version 1.0.2, `https://scikit-learn.org`.

[5]Version 3.0.5, `http://spacy.io`.

[6]Available from `https://code.google.com/archive/p/word2vec`.

[7]Available from `https://nlp.stanford.edu/projects/glove`.

[8]Available from `https://github.com/stanfordnlp/GloVe`.

window size set to 5, and other parameters left at their default settings. The following versions of the *GloVe* representation are used:

GV-P: pre-trained *GloVe* vectors,

GV-CS: custom *GloVe* vectors trained on data from the single SLR study for which a classification model is to be created,

GV-CA: custom *GloVe* vectors trained on combined data from all SLR studies.

FastText. The 300-dimensional *fastText* vectors pre-trained on the Common Crawl corpus were used.[9] For training custom *fastText* vectors, the `fasttext` Python library[10] was used (Bojanowski *et al.*, 2020), with vector dimensionality set to 100, SG training mode, and other parameters left at their default settings. The following versions of the representation are used:

FT-P: pre-trained *fastText* vectors,

FT-CS: custom *fastText* vectors trained on data from the single SLR study for which a classification model is to be created,

FT-CA: custom *fastText* vectors trained on combined data from all SLR studies.

Flair. The 2048-dimensional *Flair* embedding model pre-trained on the one-billion word news corpus distributed with the `flair` Python library[11] (Akbik *et al.*, 2021) was used, in the "fast" ("CPU-friendly") version of reduced complexity. The same library was used to train custom *Flair* embedding models, with vector dimensionality set to 100, the number of layers set to 1, and other parameters left at their default settings. In both cases stacked forward and backward models were used. The following versions of the representation are used:

FL-P: a pre-trained *Flair* embedding model,

FL-CS: a custom *Flair* embedding model trained on data from the single SLR study for which a classification model is to be created,

FL-CA: a custom *Flair* embedding model trained on combined data from all SLR studies.

BioBERT. The 768-dimensional pre-trained *BioBERT* model was applied,[12] using the `BertModel` class from the `transformers` Python library[13] (Wolf *et al.*, 2020). When applying the model to obtain text embeddings, the input text was truncated to the model's maximum sequence length of 512 tokens, which marginally affects only about 0.5% articles. Due to the computational expense of generating text embeddings from this model, a simplified de-contextualized application method of the pre-trained model was additionally considered: vectors for single words from the bag of words vocabulary were determined and then document vectors were obtained by averaging such vectors for words occurring in each document, weighted by their occurrence frequency. The following versions of the representation are therefore used:

BB-PT: a pre-trained *BioBERT* model applied to texts,

BB-PW: a pre-trained *BioBERT* model applied to single words from the bag of words vocabulary and then aggregated by averaging by their occurrence frequency.

4.2.2. Classification algorithms. The following implementations of classification algorithms described in Section 3 provided by the `scikit-learn` Python library (Pedregosa *et al.*, 2011) were used:

NB: naive Bayes classifier, classes `MultinomialNB` for the bag of words representation and `GaussianNB` for all the embedding-based representations, with the default hyperparameter setup,

LR: logistic regression, class `LogisticRegression` with the maximum number of iterations set to 200 and other hyperparameters left at default settings,

SVM: class `SVC` with radial kernel, balanced class weighting, and other hyperparameters left at default settings, with data standardization performed using the `StandardScaler` class,

RF: random forest, which is implemented as class `RandomForestClassifier` with 500 trees, balanced class weighting, and other hyperparameters left at default settings.

While some settings were manually adjusted, as specified above, no hyperparameter tuning was performed. This is because with hyperparameter tuning a separate test set or an external cross-validation loop would be needed for reliable predictive performance evaluation. Otherwise, predictive performance estimates might be overoptimistic. However, with relatively small datasets, as in the presented experimental study (and

[9]Available from `https://fasttext.cc/docs/en/english-vectors.html`.

[10]Version 0.9.2, `https://fasttext.cc`.

[11]Version 0.10, `https://github.com/flairNLP/flair`.

[12]Available from `https://huggingface.co/dmis-lab/biobert-base-cased-v1.1`.

[13]Version 4.29.2, `https://huggingface.co/docs/transformers`.

as typically encountered in systematic literature review applications) saving some data for final model evaluation after hyperparameter tuning would be problematic and have potentially harmful effects on model quality and evaluation reliability, outweighing the benefits of tuning. Mostly default settings are used therefore, which may prevent some algorithms from reaching their top possible performance level and make the reported results conservative. From a practical point of view, it may be actually even more interesting to see what level of prediction quality particular algorithms coupled with different text representations method achieve in default configurations than in carefully tuned ones.

4.2.3. BioBERT fine-tuning. The same pre-trained *BioBERT* model providing text embeddings mentioned above was also fine-tuned for the classification task using the `BertForSequenceClassification` and `Trainer` classes from the `transformers` Python library (Wolf *et al.*, 2020). Consistently with the approach adopted for the conventional algorithms, no hyperparameter tuning was performed, with the associated computational expense adding to the justification for this decision already discussed above. Two exceptions are some preliminary experiments with a custom weighted cross-entropy loss function, using class-rebalancing weights, and with model layer freezing. Based on these experiments, the training process was configured to use the custom weighted loss function, with class weights inversely proportional to the square roots of class counts, and to keep the embedding layer and the first 7 encoder layers frozen. The former was found to improve performance in comparison with the standard unweighted cross-entropy loss function, and the latter was found to save computation time without classification quality degradation. As for *BioBERT* text embeddings, the input text for training and prediction was truncated to the model's maximum sequence length of 512 tokens, which marginally affects only about 0.5% articles.

4.3. Predictive performance evaluation. Common simple classification quality measures such as the misclassification error or classification accuracy are not useful whenever classes are imbalanced or likely to have different predictability. In this article classification quality is evaluated using the ROC analysis (Egan, 1975; Fawcett, 2006), considering all possible tradeoff points between the true positive rate and the false positive rate. The former is the share of instances of the positive class which are correctly predicted to be positive and the latter is the share of instances of the negative class which are incorrectly predicted to be positive. The performance across all possible tradeoffs, corresponding to different predicted positive class probability or decision function value thresholds, can be summarized using the area under the ROC curve (AUC).

With heavily imbalanced classes and most instances being negative, the false positive rate is not sensitive to even a substantial share of incorrect positive class predictions, because the number of false positives remains small relative to the dominating negative class count. It is therefore useful to additionally consider the precision, which is the share of positive class predictions that are correct. Its tradeoff against the recall, which is another term for the true positive rate, is represented by precision-recall (PR) curves and can be summarized by the area under the PR curve (PR AUC).

For reliable prediction quality evaluation the $n \times k$-fold cross-validation procedure is applied (Arlot and Celisse, 2010), which effectively uses the available data for both model creation and evaluation. This is achieved by randomly splitting it into k equally sized subsets, each of which serves as a test set for evaluating the model created on the combined remaining subsets. This process is repeated n times for further variance reduction. The true class labels and predictions for all $n \times k$ iterations are then combined to determine ROC curves, PR curves, and the corresponding AUC values. This micro-averaging approach, suggested by Fawcett (2006) as an alternative to curve averaging, may sometimes underestimate the predictive performance if predicted probabilities or decision function values are obtained for different data folds and are not calibrated with one another (Forman and Scholz, 2010). The effect is usually small and may be in fact considered desirable in the case of text classification for systematic literature reviews, since it penalizes the inconsistency of class probabilities or decision values, which would make it problematic to set the classification threshold for the relevant class. The cross-validation procedure uses $k = 10$ folds and $n = 5$ repeats.

4.4. Results. Two major types of results are presented in this section:

text representation utility: assessing the utility of different text representation methods for particular classification algorithms, based on aggregated predictive performance over all datasets,

classification quality comparison: comparing the quality of predictions obtained by particular classification algorithms coupled with their respective best text representation methods on each dataset, as well as by a pre-trained *BioBERT* model fine-tuned for classification.

4.4.1. Text representation utility. Due to the large number of dataset, text representation, and classification

Table 3. Average ROC AUC and PR AUC values over all datasets.

(a) ROC AUC

	NB	LR	SVM	RF
BOW-TF	0.82	0.79	0.81	0.84
BOW-TFIDF	0.77	0.84	0.81	0.83
BOW-L-TF	0.82	0.79	0.81	0.84
BOW-L-TFIDF	0.77	0.84	0.81	0.83
W2V-P	0.71	0.77	0.81	0.80
W2V-CS	0.70	0.79	0.76	0.75
W2V-CA	0.76	0.77	0.81	0.81
D2V-CS	0.77	0.78	0.80	0.80
D2V-CA	0.77	0.75	0.79	0.78
GV-P	0.73	0.77	0.82	0.80
GV-CS	0.76	0.76	0.80	0.80
GV-CA	0.76	0.76	0.82	0.81
FT-P	0.78	0.75	0.83	0.81
FT-CS	0.77	0.76	0.83	0.82
FT-CA	0.79	0.77	0.84	0.83
FL-P	0.68	0.74	0.81	0.77
FL-CS	0.71	0.69	0.81	0.78
FL-CA	0.69	0.67	0.78	0.76
BB-PT	0.76	0.81	0.83	0.81
BB-PW	0.71	0.75	0.80	0.79

(b) PR AUC

	NB	LR	SVM	RF
BOW-TF	0.50	0.47	0.50	0.55
BOW-TFIDF	0.43	0.53	0.51	0.53
BOW-L-TF	0.50	0.47	0.49	0.55
BOW-L-TFIDF	0.43	0.53	0.51	0.54
W2V-P	0.32	0.42	0.50	0.47
W2V-CS	0.32	0.45	0.42	0.40
W2V-CA	0.39	0.42	0.50	0.49
D2V-CS	0.41	0.41	0.47	0.44
D2V-CA	0.42	0.38	0.47	0.45
GV-P	0.35	0.42	0.50	0.48
GV-CS	0.39	0.42	0.48	0.48
GV-CA	0.39	0.41	0.50	0.49
FT-P	0.42	0.37	0.54	0.50
FT-CS	0.42	0.39	0.52	0.50
FT-CA	0.45	0.41	0.55	0.53
FL-P	0.29	0.37	0.51	0.46
FL-CS	0.35	0.32	0.49	0.46
FL-CA	0.32	0.29	0.47	0.42
BB-PT	0.38	0.49	0.54	0.49
BB-PW	0.32	0.41	0.48	0.46

algorithm combinations only aggregated summary results over all 15 datasets are presented here. They not only take up less space, but are also easier to interpret and lead to clearer conclusions.

Table 3 presents ROC AUC and PR AUC values for all text representation and algorithm combinations, averaged over all datasets.

The following observations can be made based on these results:

- the term-frequency bag of words representation is the most useful overall, there is no improvement due to lemmatization, and using TF-IDF gives mixed results (better predictions for logistic regression, worse predictions for the naive Bayes classifier, and no significant effect for the other algorithms);
- the custom *fastText* model trained on all combined datasets appears the most useful embedding-based representation and approaches the performance level of the bag of words representations;
- custom *word2vec* vectors trained on a single dataset and *Flair* embeddings are less useful than the other embedding-based representations;
- despite the refinement and computational expense of the *BioBERT* model, the resulting text representations appear to have no major advantages over simpler and more efficient alternatives such as *fastText* or *GloVe*;
- the naive Bayes classifier works best with the bag of words representations and is less effective when using embedding-based representations;
- the SVM and random forest algorithms are the most effective overall, usually outperforming the other two.

To more reliably determine which text representations work best with particular classification algorithms, for each algorithm a 1000-sample bootstrap statistical significance test was performed to determine for all pairs of text representation methods whether one of them significantly outperformed the other on a particular dataset with respect to the area under the PR curve. Based on the results of these tests, Table 4(a) presents the ranking of text representations for each of the classification algorithms used in the experiments (corresponding to table columns), according to the number of times each representation significantly outperformed any other representation. The higher a text representation method appears in the table column corresponding to a given algorithm, the more frequently it performed significantly better than another representation method with this algorithm. The following observations can be made:

Table 4. Rankings of text representations and classification algorithms.

(a) ranking of text representations for particular classification algorithms

NB	LR	SVM	RF
BOW-TF	BOW-L-TFIDF	FT-CA	BOW-L-TF
BOW-L-TF	BOW-TFIDF	FT-P	BOW-TF
FT-CA	BB-PT	BB-PT	BOW-TFIDF
FT-P	BOW-TF	FT-CS	BOW-L-TFIDF
FT-CS	W2V-CS	GV-CA	FT-CA
BOW-TFIDF	BOW-L-TF	BOW-TFIDF	FT-P
BOW-L-TFIDF	W2V-CA	GV-P	FT-CS
D2V-CS	GV-CA	BOW-L-TFIDF	GV-CA
D2V-CA	GV-CS	FL-P	BB-PT
GV-CS	FT-CA	BOW-TF	W2V-CA
GV-CA	GV-P	W2V-CA	GV-CS
W2V-CA	D2V-CS	BOW-L-TF	GV-P
BB-PT	W2V-P	W2V-P	W2V-P
FL-CS	FT-CS	FL-CS	BB-PW
GV-P	BB-PW	BB-PW	D2V-CA
W2V-CS	D2V-CA	GV-CS	FL-CS
W2V-P	FT-P	D2V-CA	FL-P
FL-CA	FL-P	FL-CA	D2V-CS
BB-PW	FL-CS	D2V-CS	W2V-CS
FL-P	FL-CA	W2V-CS	FL-CA

(b) ranking of classification algorithms for particular text representations

BOW-TF	RF	SVM	NB	LR
BOW-TFIDF	LR	RF	SVM	NB
BOW-L-TF	RF	SVM	NB	LR
BOW-L-TFIDF	LR	RF	SVM	NB
W2V-P	SVM	RF	LR	NB
W2V-CS	LR	SVM	RF	NB
W2V-CA	SVM	RF	LR	NB
D2V-CS	SVM	RF	LR	NB
D2V-CA	SVM	RF	NB	LR
GV-P	SVM	RF	LR	NB
GV-CS	SVM	RF	LR	NB
GV-CA	SVM	RF	LR	NB
FT-P	SVM	RF	NB	LR
FT-CS	SVM	RF	NB	LR
FT-CA	SVM	RF	NB	LR
FL-P	SVM	RF	LR	NB
FL-CS	SVM	RF	NB	LR
FL-CA	SVM	RF	NB	LR
BB-PT	SVM	RF	LR	NB
BB-PW	SVM	RF	LR	NB

- the bag of words representations are always among the most successful representations for all algorithms except for SVM;
- the basic term-frequency variant of bag of words usually works best except for logistic regression, where the TF-IDF variant is usually superior;
- the custom *fastText* representation trained on all combined datasets is the most useful embedding-based representations for all the algorithms except for logistic regression, where the *BioBERT* model applied to texts is usually superior.

Similarly, to determine which classification algorithms work best with particular text representation methods, for each representation a 1000-sample bootstrap statistical significance test was performed to determine for all pairs of classification algorithms whether one of them significantly outperformed the other on a particular dataset with respect to the area under the PR curve. Based on the results of these tests, Table 4(b) presents the ranking of classification algorithms for each of the text representations used in the experiments (corresponding to table rows), according to the number of times each algorithm significantly outperformed any other algorithm. The more to the left a classification algorithms appears in the table row corresponding to a given text representation, the more frequently it performed significantly better than another algorithm with this representation. The following observations can be made:

- the random forest algorithm is the most successful for the term frequency bag of words representation;
- the SVM algorithm is the most successful for most of the embedding-based representations;
- the naive Bayes classifier cannot significantly outperform the random forest and SVM algorithms, but it tends to work better than logistic regression with the term frequency bag of words representation and with the *fastText* embeddings;
- logistic regression works well for the TF-IDF version of bag of words and for some types of embeddings (in particular, *word2vec*, *GloVe* and *BioBERT*).

It can be interesting to verify to what extent the predictions of particular classification algorithms change when switching text representation methods. Figure 1 provides some insight into that by presenting correlation plots for predicted positive class probabilities (for the naive Bayes classifier, logistic regression, and random forest algorithms) or decision function values (for the SVM algorithm) obtained by each algorithm when used with different text representations methods. It can be observed that:

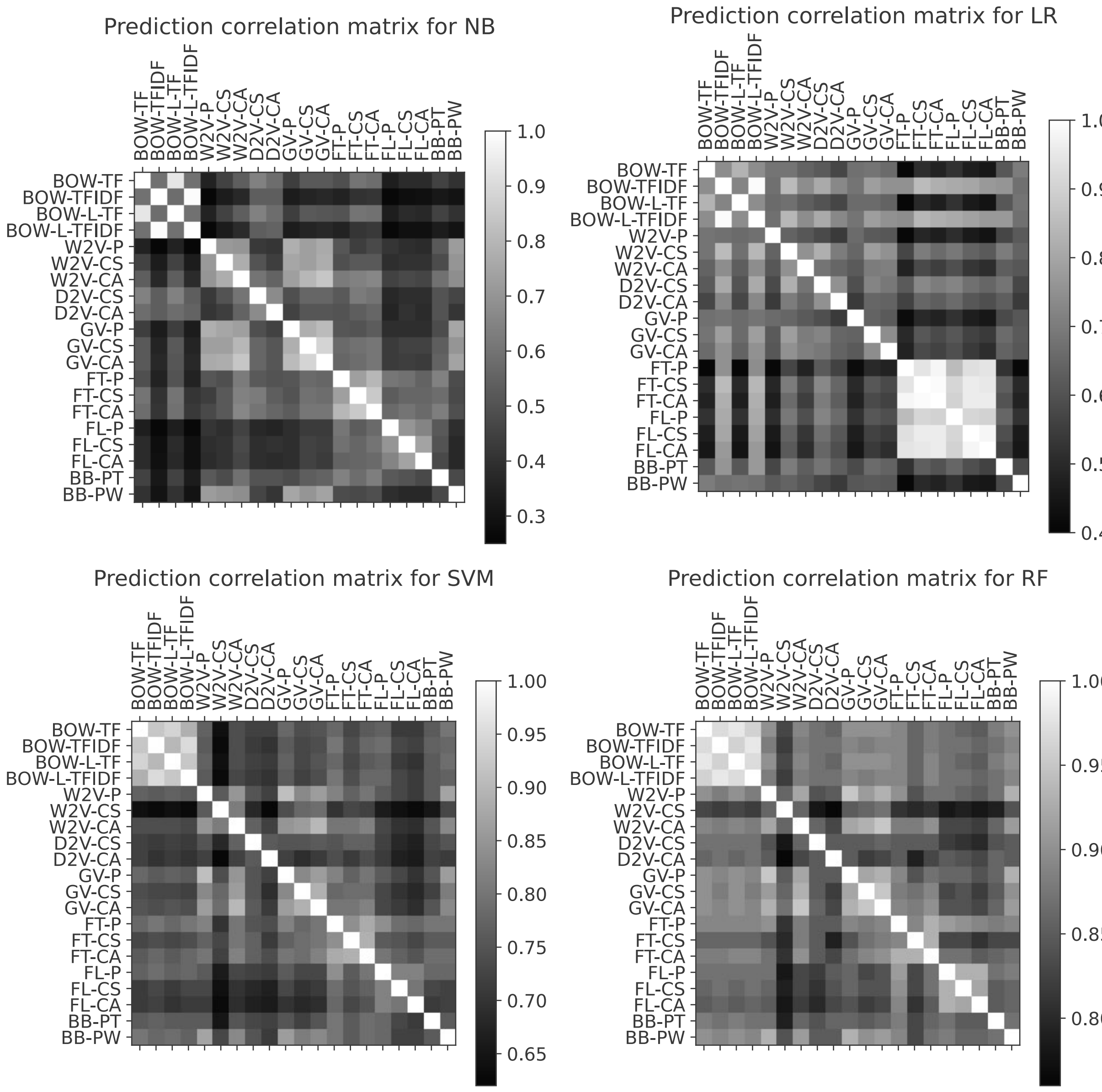

Fig. 1. Correlations between predictions obtained using different text representation methods.

- as to be expected, the bag of words variants with and without lemmatization yield very highly correlated predictions;
- higher correlations are observed between the predictions obtained using the *word2cec* and *GloVe* representations;
- for logistic regression, the predictions obtained using the TF-IDF bag of words representations have higher correlations with the predictions of the embedding-based representations than those obtained using TF bag of words;
- the logistic regression predictions obtained using the *fastText* and *Flair* embeddings are highly correlated;
- the overall level of prediction correlations is the highest for the random forest algorithm (which can be therefore considered the most representation-insensitive) and the lowest for the naive Bayes and logistic regression algorithms (which can be therefore considered the most representation-sensitive).

4.4.2. Classification quality comparison. For a more detailed view of the predictive performance, Table 5

Table 5. ROC AUC and PR AUC values for particular datasets obtained by each algorithm with the best text representation.

(a) ROC AUC

	NB	LR	SVM	RF	BB
ACEInhibitors	*0.82*	0.85	0.85	0.83	**0.88**
ADHD	*0.92*	0.93	0.93	**0.94**	**0.94**
Antihistamines	0.75	0.77	*0.72*	**0.79**	0.73
AtypicalAntipsychotics	*0.79*	0.80	0.81	*0.79*	**0.82**
BetaBlockers	*0.79*	0.83	**0.84**	0.80	0.81
CalciumChannelBlockers	*0.80*	0.83	**0.85**	0.84	**0.85**
Estrogens	**0.87**	**0.87**	0.86	0.86	*0.81*
NSAIDS	0.89	0.89	**0.90**	**0.90**	*0.88*
Opiods	0.84	0.86	**0.89**	0.85	*0.81*
OralHypoglycemics	*0.74*	0.75	0.75	0.76	**0.77**
ProtonPumpInhibitors	*0.77*	0.79	0.80	**0.81**	0.79
SkeletalMuscleRelaxants	0.82	**0.87**	0.85	0.84	*0.76*
Statins	*0.81*	0.84	**0.85**	0.82	*0.81*
Triptans	*0.86*	**0.89**	**0.89**	**0.89**	0.88
UrinaryIncontinence	0.84	0.85	**0.86**	0.85	*0.83*
Average	*0.82*	**0.84**	**0.84**	**0.84**	*0.82*

(b) PR AUC

	NB	LR	SVM	RF	BB
ACEInhibitors	*0.33*	0.40	0.41	0.40	**0.45**
ADHD	*0.48*	0.56	0.60	0.63	**0.64**
Antihistamines	*0.53*	0.61	0.59	**0.64**	0.54
AtypicalAntipsychotics	*0.66*	*0.66*	0.69	0.67	**0.72**
BetaBlockers	*0.44*	**0.51**	0.49	0.45	0.50
CalciumChannelBlockers	*0.55*	0.60	0.62	**0.66**	0.65
Estrogens	0.64	**0.67**	**0.67**	0.64	*0.50*
NSAIDS	*0.69*	0.74	0.74	**0.77**	0.72
Opiods	*0.17*	*0.17*	0.23	0.24	**0.32**
OralHypoglycemics	*0.58*	**0.61**	*0.58*	**0.61**	0.59
ProtonPumpInhibitors	*0.49*	0.54	0.51	**0.55**	0.51
SkeletalMuscleRelaxants	0.25	0.24	0.30	**0.35**	*0.10*
Statins	*0.33*	0.38	**0.41**	0.36	0.35
Triptans	*0.72*	0.76	0.75	0.78	**0.79**
UrinaryIncontinence	0.59	*0.56*	**0.65**	0.57	0.62
Average	*0.50*	0.53	**0.55**	**0.55**	0.53

presents the area under the ROC and PR curves obtained for each dataset by each classification algorithm coupled with its overall best text representation, as determined by the ranking shown in Table 4(a) (i.e., the overall top-ranked representation for a given algorithm is selected and used for all datasets). This is also where the results obtained by the pre-trained *BioBERT* model fine-tuned for classification are presented in the last column. The bottom row of each table presents the average over all datasets. The maximum value in each row is marked by a bold font and the minimum value in each row is marked by an italic font. The following observations can be made based on these results:

- when coupled with the best text representation methods, all the conventional algorithms deliver a similar level of prediction quality, with the SVM and random forest algorithms slightly better on the average than the naive Bayes and linear regression algorithms;
- no single algorithm is the best or the worst on all datasets;
- the random forest, SVM, and fine-tuned *BioBERT* algorithms are the most frequent winners, and the naive Bayes classifier is the most frequent loser;
- the average predictive performance of the fine-tuned *BioBERT* model is similar to that of the conventional algorithms, although for some datasets it differs substantially, e.g., it is clearly better for the *Opiods* datasets, but much worse for the *SkeletalMuscleRelaxants* dataset (interestingly, these two datasets are of similar size and imbalance ratio).

It is interesting to observe how the prediction quality on particular datasets is related to their properties presented in Table 2. Table 6 demonstrates the linear correlation values between the PR AUC of each algorithm and the size as well as relevant class percentage. It can be seen that the predictive performance of all the algorithms is quite strongly correlated positively with the relevant class percentage, which is clearly to be expected. What may appear somewhat more surprising is the negative correlation with the dataset size. The contradiction with the expectation, based on both common sense and machine learning theory, that more data permits better models, is only apparent, though, since this expectation would be justified only if the number of training instances were increased for the same classification task, while keeping the test set size unchanged. Here different datasets correspond to different classification tasks, the training and test data are sized proportionally under the k-fold cross-validation procedure, and bigger datasets are likely to contain more articles that are hard to correctly separate.

Finally, to visualize the predictive performance possible to obtain on the “easiest” and “hardest” data, Figures 2 and 3 present the ROC and PR curves obtained using the two most successful classification algorithms, SVM and random forest, with selected text representation methods, on the *SkeletalMuscleRelaxants* and *Triptans* datasets. These are the two datasets with the highest and lowest level of class imbalance, respectively.

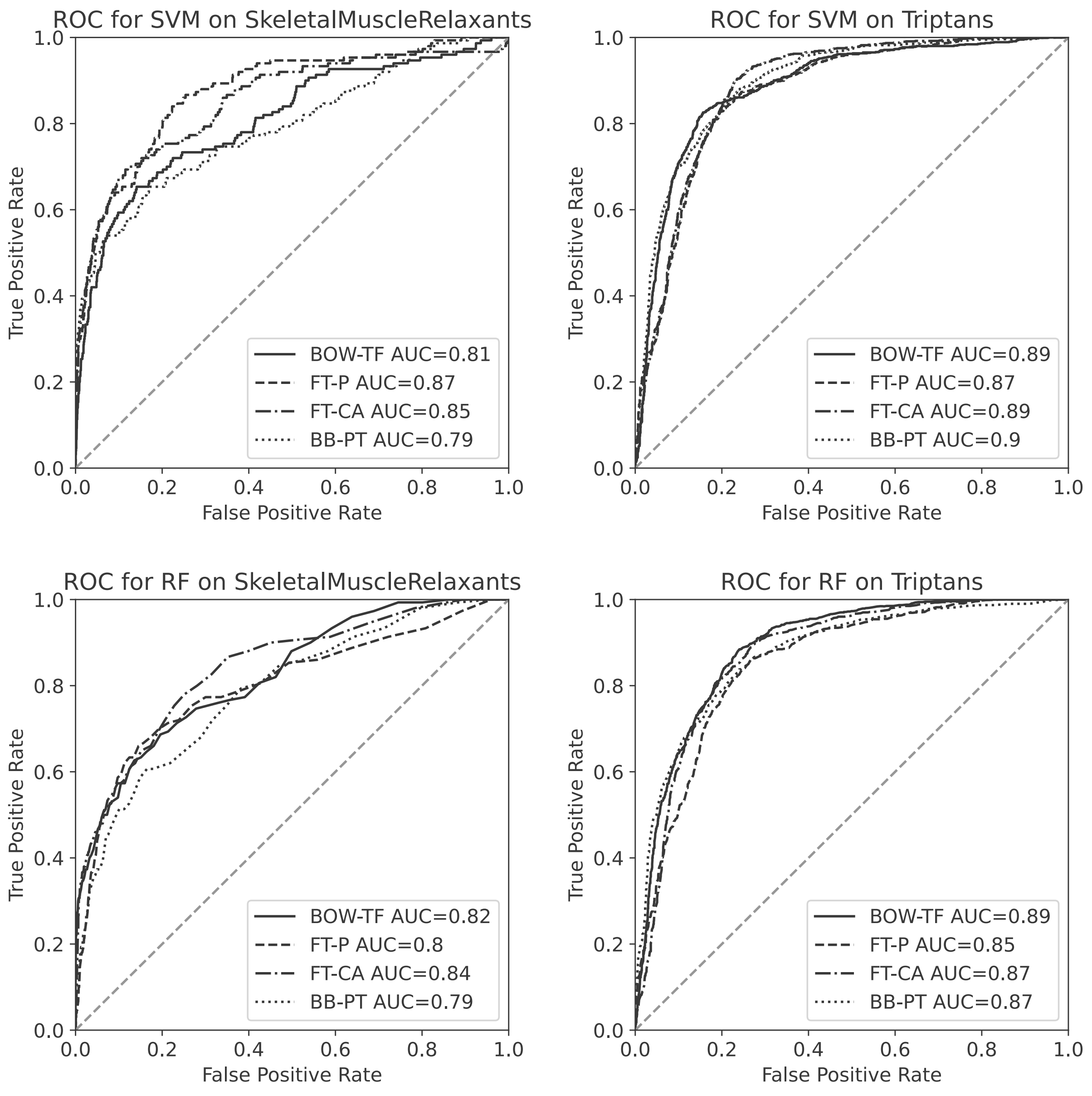

Fig. 2. Selected ROC curves.

Not surprisingly, precision-recall curves differ between the two datasets much more substantially than ROC curves. This confirms that class imbalance makes it a serious challenge for classification models to maintain a reasonable level of precision for useful values of recall. The results on the dataset with just above 2% relevant articles can be hardly considered satisfactory, while those on the dataset with more that one-third of relevant articles are quite good. The performance differences between particular text representation methods are also more pronounced on the more imbalanced data.

4.5. Discussion. Unfortunately no single classification algorithm and text representation combination yields the best models for all data. However, based on the results obtained on 15 datasets, corresponding to different SLR studies and exhibiting varying size and class imbalance level, it is possible to derive some recommendations. While the bag of words representation makes it possible to achieve the best level of prediction quality, some embedding-based representation reach nearly the same predictive performance and offer other noteworthy advantages. In particular, *fastText* embeddings, obtained using either a general purpose pre-trained model or,

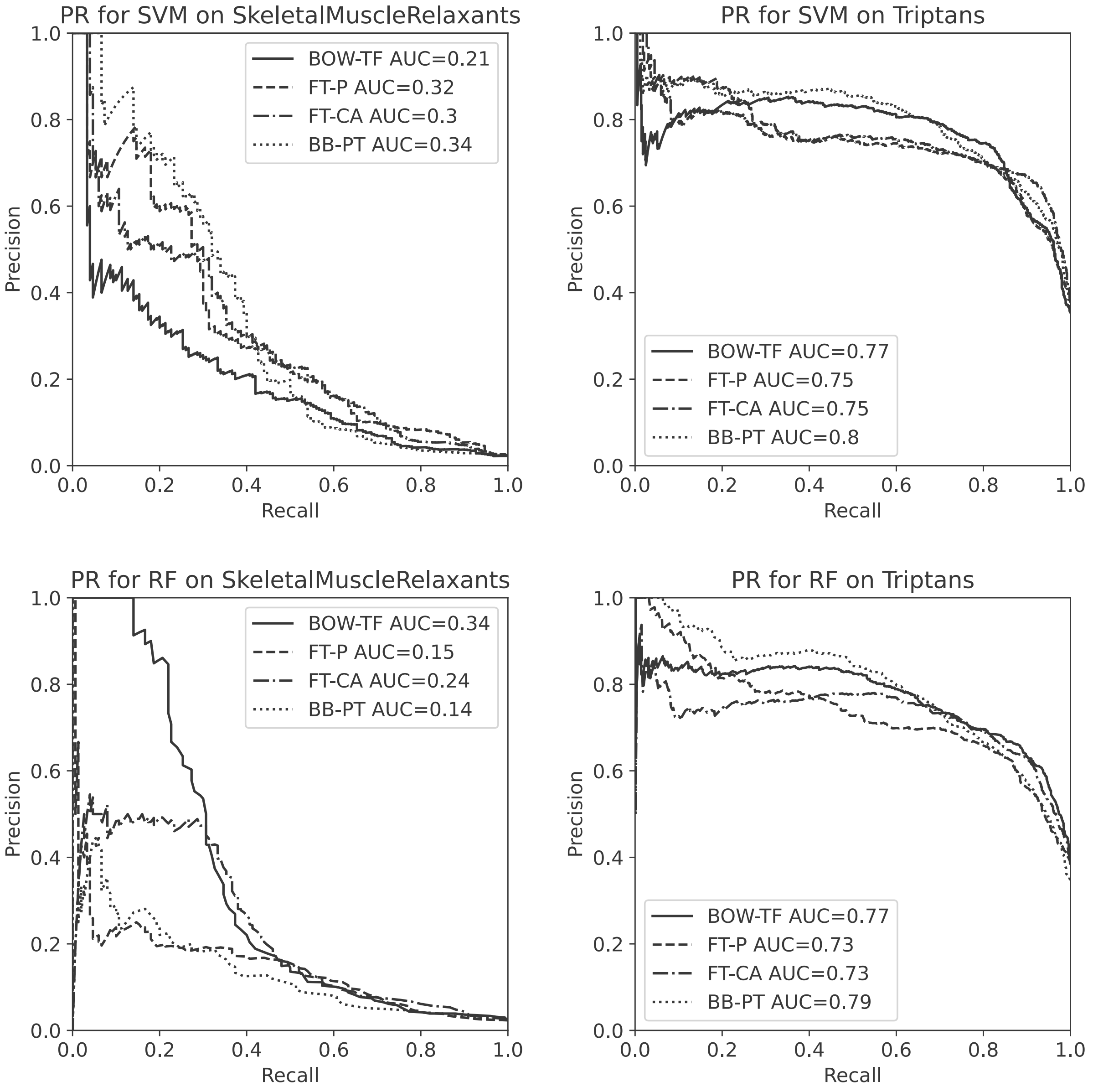

Fig. 3. Selected PR curves.

preferably, a custom domain-specific model, turned out particularly useful, making it possible to achieve a high level of classification quality. It worked particularly well with the SVM and random forest algorithms, made it possible to keep input dimensionality relatively low, and did not incur a high computational expense. The SVM algorithm was found to be the most effective for all of the embedding-based representations, although the random forest algorithm can be considered more universal and safe to use, since it delivered models of nearly the same quality for several embedding-based representations and performed better with bag of words. Its relatively low sensitivity to the choice of text representation, confirmed by a high level of prediction correlations obtained with different representations, is an interesting and useful property, which adds to its low sensitivity to hyperparameter settings and makes it particularly easy to use for text classification. For the other algorithms, more sensitive to the choice of text representation, it is possible that combining representations for which predictions are of similar quality but not very highly correlated would give some improvement. However, the weakest prediction correlations usually tend to occur for representation pairs of which one is clearly superior to the other and this does

Table 6. Correlations between PR AUC values and the dataset size as well as relevant class percentage.

	NB	LR	SVM	RF	BB
Size	–0.74	–0.69	–0.72	–0.77	–0.56
Relevant %	0.91	0.88	0.86	0.86	0.81

not appear to be a very promising direction of further exploration.

Interestingly, *BioBERT*—a state-of-the art contextual bidirectional neural language model trained on biomedical text corpora—did not provide a more useful representation for text classification, whereas requiring much more computation time. Similarly as the text embeddings obtained using *BioBERT* were comparably useful to those from simpler and more efficient embedding models, the *BioBERT* model fine-tuned as a classifier delivered predictions of similar quality to that of conventional algorithms when they were coupled with their respective best text representation methods. One possible reason of the fine-tuned *BioBERT* not showing its full potential may be the relatively small size and high level of class imbalance of the datasets. It is not unlikely that with some hyperparameter tuning, different class weights for a custom loss function or a resampling scheme increasing sensitivity to the minority class, or some data augmentation the *BioBERT* model would deliver improved prediction quality, exceeding that of the conventional algorithms, but examining this possibility is beyond the objectives and scope of this article.

5. Conclusion

The results of this work confirm that the choice of text representation is essential for successful text classification. While not at all surprising, this has not apparently received as much recognition as it deserves. Several studies on text classification focus much more on classification algorithms and their setup than on the type of vector representation to which the text is transformed to make these algorithms applicable.

Indeed, according to the presented experimental study, each of the four applied classification algorithms is usually capable of delivering top prediction quality when coupled with an appropriate text representation, although some of them (SVM, random forest) work well with a broader range of text representations than some others (naive Bayes, logistic regression). However, predictive performance differences for the same classification algorithm used with different text representation may be more substantial than differences between different algorithms. While the ubiquitous bag of words representation makes it possible to achieve the best level of prediction quality, it has some serious competition among representations based on word or text embeddings, with the *fastText* representation being particularly promising because of its good predictive performance and low computational expense.

Interestingly, the pre-trained *BioBERT* model fine-tuned for classification performed on par with conventional classification algorithms with the best text representation methods. This may appear surprising and contradict the popular belief in the universal superiority of deep learning language models. These models achieve indeed spectacularly good performance in a variety of complex natural language processing tasks, such as summarization, translation, question answering, or topic detection (Babić *et al.*, 2020). They can be also successfully fine-tuned for classification (Zymkowski *et al.*, 2022). Nevertheless, one should not take it for granted they would necessarily always beat conventional algorithms in tasks to which the latter are well suited. When they achieve similar predictive performance, conventional algorithms would be clearly preferred due to much lower computational demands.

Despite the broad scope of the experimental study reported in this article it leaves some interesting issues postponed for future work. Two continuation directions are particularly noteworthy. First, since the hardness of text classification is often related to class imbalance and it is natural for systematic literature review data to have heavily imbalanced classes, it would make sense to investigate the utility of different imbalance compensation techniques when combined with selected text representation methods and classification algorithms. In the reported experiments class rebalancing weights were used, but data resampling approaches, including generating synthetic minority class instances (Chawla *et al.*, 2002; Menardi and Torelli, 2014; Koziarski and Woźniak, 2017), might sometimes work better. Second, since text classification models for systematic literature reviews usually have to be created using a small set of labeled training instances, it would be desirable to verify the utility of particular classification algorithms and text representations when applied as part of training schemes specifically designed for model creation with few class labels, such as active learning (Cohn *et al.*, 1994) or semi-supervised learning (Zhu and Goldberg, 2009).

Acknowledgment

This work has been supported by Norway Grants under the contract no. NOR/POLNOR/REFSA/0059/2019-00.

References

Aggarwal, C.C. and Zhai, C.-X. (Eds) (2012). *Mining Text Data*, Springer, New York.

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S. and Vollgraf, R. (2019). FLAIR: An easy-to-use framework for state-of-the-art NLP, *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), Stroudsburg, USA*, pp. 54–59.

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S. and Vollgraf, R. (2021). *Flair: A Very Simple Framework for State-of-the-Art NLP*, Version 0.10, `https://github.com/flairNLP/flair`.

Akbik, A., Blythe, D. and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling, *Proceedings of the 27th International Conference on Computational Linguistics, COLING-2018, Santa Fe, USA*, pp. 1638–1649.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection, *Statistics Surveys* **4**: 40–79.

Babić, K., Martincic-Ipsic, S. and Meštrović, A. (2020). Survey of neural text representation models, *Information* **11**(11): 511.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2016). Enriching word vectors with subword information, *arXiv:* 1607.04606.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2020). *fastText: Library for Efficient Text Classification and Representation Learning*, Version 0.9.2, `https://fasttext.cc`.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M. and Kasneci, G. (2022). Deep neural networks and tabular data: A survey, *arXiv:* 2110.01889.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(1): 5–32.

Chawla, N.V., Bowyer, K. W. Hall, L.O. and Kegelmeyer, W.P. (2002). SMOTE: Synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research* **16**: 321–357.

Cichosz, P. (2018). A case study in text mining of discussion forum posts: Classification with bag of words and global vectors, *International Journal of Applied Mathematics and Computer Science* **28**(4): 787–801, DOI: 10.2478/amcs-2018-0060.

Cohen, A.M., Hersh, W.R., Peterson, K. and Yen, P.-Y. (2006). Reducing workload in systematic review preparation using automated citation classification, *Journal of the American Medical Informatics Association* **13**(2): 206–219.

Cohn, D., Atlas, L. and Ladner, R. (1994). Improving generalization with active learning, *Machine Learning* **15**(2): 201–221.

Cortes, C. and Vapnik, V.N. (1995). Support-vector networks, *Machine Learning* **20**(3): 273–297.

Dařena, F. and Žižka, J. (2017). Ensembles of classifiers for parallel categorization of large number of text documents expressing opinions, *Journal of Applied Economic Sciences* **12**(1): 25–35.

Deb, S. and Chanda, A.K. (2022). Comparative analysis of contextual and context-free embeddings in disaster prediction from Twitter data, *Machine Learning with Applications* **7**: 100253.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding, *Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT-2019, Minneapolis, USA*, pp. 4171–4186.

Dumais, S.T., Platt, J.C., Heckerman, D. and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization, *Proceedings of the 17th International Conference on Information and Knowledge Management, CIKM-98, Bethesda, USA*, pp. 148–155.

Egan, J.P. (1975). *Signal Detection Theory and ROC Analysis*, Academic Press, New York.

Fawcett, T. (2006). An introduction to ROC analysis, *Pattern Recognition Letters* **27**(8): 861–874.

Forman, G. (2003). An extensive empirical study of feature selection measures for text classification, *Journal of Machine Learning Research* **3**: 1289–1305.

Forman, G. and Scholz, M. (2010). Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement, *ACM SIGKDD Explorations Newsletter* **12**(1): 49–57.

García Adeva, J.J., Pikatza Atxaa, J.M., Ubeda Carrillo, M. and Ansuategi Zengotitabengoa, E. (2014). Automatic text classification to support systematic reviews in medicine, *Expert Systems with Applications* **41**(4): 1498–1508.

Graves, A. (2013). Generating sequences with recurrent neural networks, *arXiv:* 1308.0850.

Hamel, L.H. (2009). *Knowledge Discovery with Support Vector Machines*, Wiley, Hoboken.

Hassan, S., Mihalcea, R. and Banea, C. (2007). Random-walk term weighting for improved text classification, *Proceedings of the 1st IEEE International Conference on Semantic Computing, ICSC-2007, Irvine, USA*, pp. 53–60.

Helaskar, M.N. and Sonawane, S.S. (2019). Text classification using word embeddings, *Proceedings of the 5th International Conference on Computing, Communication, Control, and Automation, ICCUBEA-2019, New York, USA*, pp. 1–4.

Hilbe, J.M. (2009). *Logistic Regression Models*, Chapman and Hall, Boca Raton.

Honnibal, M., Montani, I., Van Landeghem, S. and Boyd, A. (2021). *spaCy: Industrial-Strength Natural Language Processing in Python*, `http://spacy.io`.

Ji, X., Ritter, A. and Yen, P.-Y. (2017). Using ontology-based semantic similarity to facilitate the article screening process for systematic reviews, *Journal of Biomedical Informatics* **69**: 33–42.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features, *Proceedings of the 10th European Conference on Machine Learning, ECML-98, Chemnitz, Germany*, pp. 137–142.

Joachims, T. (2002). *Learning to Classify Text by Support Vector Machines: Methods, Theory, and Algorithms*, Springer, New York.

Jonnalagadda, S. and Petitti, D. (2013). A new iterative method to reduce workload in systematic review process, *International Journal of Computational Biology and Drug Design* **6**(1–2): 5–17.

Kaibi, I., Nfaoui, E.H. and Satori, H. (2019). A comparative evaluation of word embeddings techniques for Twitter sentiment analysis, *Proceedings of the 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems, WITS-2019, Fez, Morocco*, pp. 1–4.

Khabsa, M., Elmagarmid, A., Ilyas, I., Hammady, H. and Ouzzani, M. (2016). Learning to identify relevant studies for systematic reviews using random forest and external information, *Machine Learning* **102**(3): 465–482.

Koprinska, I., Poon, J., Clark, J. and Chan, J. (2007). Learning to classify e-mail, *Information Sciences: An International Journal* **177**(10): 2167–2187.

Koziarski, M. and Woźniak, M. (2017). CCR: A combined cleaning and resampling algorithm for imbalanced data classification, *International Journal of Applied Mathematics and Computer Science* **27**(4): 727–736, DOI: 10.1515/amcs-2017-0050.

Le, Q.V. and Mikolov, T. (2014). Distributed representations of sentences and documents, *Proceedings of the 31st International Conference on Machine Learning, ICML-2014, Beijing, China*, pp. 1188–1196.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S.and So, C.H. and Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics* **36**(4): 1234–1240.

Lewis, D.D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval, *Proceedings of the 10th European Conference on Machine Learning, ECML-98, Chemnitz, Germany*, pp. 4–15.

Matwin, S., Kouznetsov, A., Inkpen, D., Frunza, O. and O'Blenis, P. (2010). A new algorithm for reducing the workload of experts in performing systematic reviews, *Journal of the American Medical Informatics Association* **17**(4): 446–453.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive Bayes text classification, *Proceedings of the AAAI/ICML-98 Workshop on Learning for Text Categorization, Madison, USA*, pp. 41–48.

Menardi, G. and Torelli, N. (2014). Training and assessing classification rules with imbalanced data, *Data Mining and Knowledge Discovery* **28**(1): 92–122.

Mikolov, T., Chen, K., Corrado, G.S. and Dean, J. (2013). Efficient estimation of word representations in vector space, *arXiv:* 1301.3781.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics, *Cognitive Science* **34**(8): 1388–1429.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Pennington, J., Socher, R. and Manning, C.D. (2014). GloVe: Global vectors for word representation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP-2014, Doha, Qatar*, pp. 1532–1543.

Platt, J.C. (1998). Fast training of support vector machines using sequential minimal optimization, *in* B. Schölkopf *et al.* (Eds), *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge, pp. 185–208.

Radovanović, M. and Ivanović, M. (2008). Text mining: Approaches and applications, *Novi Sad Journal of Mathematics* **38**(3): 227–234.

Řehůřek (2021). *Gensim: Topic Modeling for Humans*, Version 4.0.1, `https://radimrehurek.com/gensim`.

Řehůřek, V. and Sojka, P. (2010). Software framework for topic modelling with large corpora, *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta*, pp. 45–50.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B.B., Chen, X. and Wang, X. (2020). A survey of deep active learning, *ACM Computing Surveys* **54**(9): 1–40.

Rios, G. and Zha, H. (2004). Exploring support vector machines and random forests for spam detection, *Proceedings of the 1st Conference on Email and Anti Spam, CEAS-2004, Moutain View, USA*, pp. 284–292.

Salton, G. and Buckley, C. (1988). Term weighting approaches in automatic text retrieval, *Information Processing and Management* **24**(5): 513–523.

Szymański, J. (2014). Comparative analysis of text representation methods using classification, *Cybernetics and Systems* **45**(2): 180–199.

van den Bulk, L.M., Bouzembrak, Y., Gavai, A., Liu, N., van den Heuvel, L.J. and Marvin, H.J.P. (2022). Automatic classification of literature in systematic reviews on food safety using machine learning, *Current Research in Food Science* **5**: 84–95.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need, *Advances in Neural Information Processing Systems, NIPS-2017, Long Beach, USA*, pp. 6000–6010.

Wang, C., Nulty, P. and Lillis, D. (2020). A comparative study on word embeddings in deep learning for text classification, *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, NLPIR-2020, Seoul, Korea*, pp. 37–46.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q. and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, (online).

Xue, D. and Li, F. (2015). Research of text categorization model based on random forests, *2015 IEEE International Conference on Computational Intelligence and Communication Technology, CICT-2015, Ghaziabad, India*, pp. 173–176.

Yang, Y. and Pedersen, J. (1997). A comparative study on feature selection in text categorization, *Proceedings of the 14th International Conference on Machine Learning, ICML-97, Nashville, USA*, pp. 412–420.

Yessenalina, A. and Cardie, C. (2011). Compositional matrix-space models for sentiment analysis, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP-2011, Edinburgh, UK*, pp. 172–182.

Zhu, X. and Goldberg, A. (2009). *Introduction to Semi-Supervised Learning*, Morgan & Claypool, San Rafael,.

Zymkowski, T., Szymański, J., Sobecki, A., Drozda, P., Szałapak, K., Komar-Komarowski, K. and Scherer, R. (2022). Short texts representations for legal domain classification, *Proceedings of the 21st International Conference on Artificial Intelligence and Soft Computing, ICAISC-2022, Zakopane, Poland*, pp. 105–114.

Paweł Cichosz received his MSc and PhD degrees in computer science from the Warsaw University of Technology in 1994 and 1998, respectively. He is an assistant professor at the Institute of Computer Science there. His areas of research interests include machine learning, data mining, natural language processing, and artificial intelligence. He has also practical experience in applied data science projects.

Received: 19 April 2023
Revised: 19 July 2023
Accepted: 28 August 2023