Text Classification: A Comprehensive Survey of Methods, Applications, and Future Directions

(Authors Details)
Sanjay Nakharu Prasad Kumar
George Washington University, USA

Abstract:

Text classification stands as a fundamental task in natural language processing, involving the automated assignment of predefined categories to textual documents, sentences, or phrases. This comprehensive survey examines the evolution of text classification methodologies from traditional machine learning approaches through deep learning innovations to contemporary transformer-based architectures. We analyze the progression from feature-engineered methods like Naïve Bayes and Support Vector Machines, through representation learning with CNNs and LSTMs, to pre-trained language models including BERT, RoBERTa, and GPT variants. Our analysis encompasses benchmark datasets, evaluation metrics, and quantitative performance comparisons across methodologies. We explore emerging paradigms including prompt-based learning, few-shot classification, and multilingual adaptation while addressing critical challenges in interpretability, fairness, and computational efficiency. The survey provides practitioners with decision frameworks for selecting appropriate approaches based on task requirements, data availability, and resource constraints. We conclude by identifying open research questions and future directions that will shape the next generation of text classification systems.

Keywords: Text Classification, Natural Language Processing, Machine Learning, Deep Learning, Transformers, BERT, Neural Networks, Sentiment Analysis, Topic

DOI: 10.21590/ijtmh.8.03.04

I. INTRODUCTION

Text classification is the task of assigning predefined labels or categories to textual documents. It is a fundamental problem in natural language processing (NLP) and information retrieval, encompassing tasks such as spam detection, sentiment analysis, topic labeling, and question answering.[1][2] The goal is to design algorithms that can automatically categorize any input text (e.g., email, review, news article) into one of several classes based on its content. As *Sebastiani* explains, text classification "is defined as learning to distinguish among a set of predefined classes using features extracted from textual documents."[3] In other words, a model is trained on labeled examples so that new, unseen texts can be accurately labeled by the learned classifier.

The importance of text classification has grown enormously in recent decades due to the explosion of digital text data. With the advent of the internet, social media, and digital publishing, the volume of textual information (blogs, reviews, news, social posts, etc.) has increased exponentially.[4][5] Organizing this data – for example by filtering spam emails, sorting news by topic, or summarizing public opinion – is critical for both users and automated systems. Early work in information retrieval recognized that automated text categorization could “alleviate the problem of information overload by locating the required information” in massive document collections.[6][7] Consequently, text classification has become a core function in many applications (e.g. search engines, recommendation systems, and enterprise data mining).

A typical text classification pipeline involves preprocessing raw text (tokenization, normalization, etc.), extracting or learning features from the text, and then applying a supervised learning algorithm to map features to labels. Classical systems rely on manually engineered features (like bag-of-words counts) and traditional classifiers, whereas more recent approaches (deep learning and transformers) learn distributed representations and decision functions end-to-end. Despite these differences, the fundamental task remains the same: use a model to predict labels for text inputs based on patterns learned from labeled data.[3][1] Text classification is also closely related to topic classification or topic spotting, where documents are categorized by topic; as Manning et al. note, “classes are usually referred to as topics, and the classification task is then called text classification, text categorization, topic classification, or topic spotting.”[8]

2. Traditional Machine Learning Methods

Early approaches to text classification treated text as high-dimensional vectors of word features and applied well-known supervised algorithms. The first step is typically feature extraction: for example, documents may be converted into bag-of-words (count) vectors or TF-IDF vectors (term frequency–inverse document frequency). These representations disregard word order but capture word presence and frequency as predictors. Standard feature processing also includes tokenization, stop-word removal, and optional stemming or lemmatization to reduce vocabulary size.[9][10]

Once features are extracted, classic classification algorithms can be used. Naive Bayes (NB) has been one of the most enduring methods for text data; it assumes word features are independent given the class label and computes class probabilities accordingly. Support Vector Machines (SVM) have also been very popular: they find a maximum-margin linear decision boundary in the feature space. Other common classifiers include decision trees, k-nearest neighbors (k-NN), and ensemble methods like random forests. For example, surveys and tutorials routinely cite Naive Bayes, SVM, and decision trees as “classic” text classifiers.[11][10] Kowalski’s review, for instance, highlights Naive Bayes, SVM, k-NN, and decision tree algorithms as foundational supervised techniques.[11]

These traditional methods generally rely on manual feature engineering but have several advantages. They are relatively fast to train and interpret and can perform well even with limited data. For instance, Naive Bayes can be implemented very efficiently and often achieves strong baselines for tasks like spam filtering. SVMs are effective for high-dimensional sparse data and have shown good performance on many text tasks. Decision trees and random forests can capture non-linear feature interactions and provide some interpretability via feature importance. However, they also have limitations: Bag-of-Words models ignore word order and context, and linear models may fail to capture complex semantics. In practice, careful feature weighting (e.g. TF-IDF) and feature selection are often applied to improve these traditional classifiers.[10][9]

3. Deep Learning Approaches

With the rise of deep learning, neural network models that automatically learn features from raw text have become dominant. These models operate on distributed representations of words (embeddings) rather than sparse vectors and can capture sequential or hierarchical patterns. Among the earliest deep architectures for text were Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), often with Long Short-Term Memory (LSTM) units to handle long-range dependencies.

Convolutional Neural Networks (CNNs): CNNs were originally proposed for image data, but they have been successfully adapted to text classification.[12] In a text-CNN, each word is first mapped to a fixed-length embedding vector, and the entire document is represented as a sequence or matrix of word vectors. Convolutional filters of various widths then slide over this sequence to detect local n-gram patterns or phrases. After convolution, pooling layers (often max pooling) select the most salient features, and a final classification layer produces the output. CNNs can capture position-invariant features (e.g., recognizing a particular phrase anywhere in the text) and are highly parallelizable. As Li et al. note, “CNNs are proposed for image classification with convolving filters that extract features; unlike RNN, CNN can apply multiple kernels simultaneously, and therefore CNNs are used for many NLP tasks including text classification.”[12] In practice, CNN-based text classifiers (like the well-known TextCNN) achieve strong results on sentiment and topic tasks by effectively learning important n-gram features.

Recurrent Neural Networks (RNNs) and LSTMs: RNNs process a sequence of inputs one token at a time, maintaining a hidden state that captures information about the sequence so far. This makes them suitable for text, which has an inherent order. A standard (Elman) RNN at step *t* updates its hidden state based on the current word embedding and the previous hidden state, effectively carrying context through time. In theory, RNNs can capture long-range dependencies, but in practice simple RNNs struggle with very long text due to vanishing or exploding gradients. LSTM units address this by introducing gated memory cells: they contain input, forget, and output gates that regulate the flow of information, allowing the network to retain or discard context as needed. This helps alleviate the vanishing gradient problem and lets the model “remember” information over many steps. Nguyen and Sudirman describe LSTM as an “improvement of RNN,” with gating mechanisms that better capture sentence context and avoid forgetting.[13]

In a typical RNN/LSTM classifier, each word is first converted to an embedding, and then the sequence of embeddings is fed through one or more RNN/LSTM layers. The final hidden state (or an aggregate of hidden states) is then passed to a softmax output layer to predict the class. Bidirectional RNNs/LSTMs (BiLSTM) are also common: one RNN processes the sequence forward and another backward, capturing both past and future context. These models excel at tasks where the order and context of words matter (e.g. sentiment depends on negation words like “not”). Despite their sequential nature, training can be slower (since RNNs process one step at a time), and they may still have difficulty with very long documents. However, deep RNNs/LSTMs with attention mechanisms remain powerful tools for many text classification problems.[14][13]

Hybrid and Other Architectures: Beyond plain CNNs and RNNs, many architectures combine both or add attention. For example, a model might use a CNN to extract local features and then feed them into an

RNN for sequential modeling. Attention mechanisms can be used on top of RNNs to weight different parts of the text differently. In general, deep neural classifiers move away from manual feature engineering: as Li et al. observe, deep learning "integrates feature engineering into the model fitting process by learning a set of nonlinear transformations that map features directly to outputs."[15][12] The trade-off is that these models often require more data and computation, but they have achieved state-of-the-art results on many benchmarks.

4. Transformer-Based Models

The most recent revolution in text classification comes from transformer-based models. Transformers are deep neural architectures relying solely on attention mechanisms, introduced by Vaswani et al. (2017). Unlike RNNs, transformers process the entire sequence in parallel using self-attention, which allows them to capture long-range dependencies efficiently. Pre-trained transformer models have become the de facto standard for many NLP tasks, including text classification.[16][17]

BERT and Its Variants: BERT (Bidirectional Encoder Representations from Transformers) was introduced by Google in 2018 and uses a masked language modeling (MLM) objective to learn deep bidirectional representations.[18] In practice, BERT and its successors are first pre-trained on massive unlabeled text and then fine-tuned on specific tasks by adding a simple classification head. BERT "significantly improves performance on NLP tasks, including text classification" when fine-tuned on task data.[17] Its deep bidirectional encoding (attending to both left and right context) allows it to produce context-sensitive embeddings that capture rich semantic information.

RoBERTa is a closely related model that tweaks the BERT recipe: it trains for longer with more data, uses dynamic masking, and drops the next-sentence-prediction task.[19] RoBERTa can be thought of as an "improved version of BERT" that shows stronger classification results when properly tuned. Other BERT-family models include ALBERT (smaller, factorized BERT) and DistilBERT (compressed BERT) designed for efficiency, and domain-specific BERT models (e.g. BioBERT for biomedical text).

GPT and Large Language Models: On the other side are autoregressive transformer models like OpenAI's GPT series. These decoder-only models are trained to predict the next token in a sequence (left-to-right). GPT-3 (2020) and GPT-4 (2023) are extremely large models with hundreds of billions of parameters, capable of generating fluent text. For classification, GPT models can be used in a few ways: they can be fine-tuned on labeled data, or more recently used in a "prompting" manner for zero/few-shot classification. The trade-off is that GPT-style LLMs tend to be very large and computationally expensive; however, they also exhibit strong "in-context learning" abilities. As Galke et al. note, decoder-only models focus on generation with remarkable in-context learning, making them "strong zero-shot and few-shot models."[16]

In practice, BERT (and its variants) and GPT are all based on transformers but serve different roles. BERT-style models (encoder-only) are typically fine-tuned discriminative classifiers, while GPT-style models (decoder-only) are often used as general-purpose generators or via prompting. Recent research shows fine-tuning BERT remains highly effective for text classification, and GPT-based classifiers can serve as an alternative especially in low-data regimes.[20][16] In any case, transformer models have dramatically advanced text classification: encoder-only models set new benchmarks, and even decoders like GPT can be applied (with prompting or fine-tuning) to classification tasks.

5. Preprocessing and Feature Engineering

Regardless of the model, most text classification systems begin with preprocessing the raw text and extracting features. Text preprocessing typically includes tokenization (splitting text into words or subword tokens), lowercasing, and removing punctuation. Stop-word removal (discarding common words like “the”, “and”) is often used to reduce noise and dimensionality. Techniques like stemming or lemmatization (reducing words to their base form) may be applied to aggregate variants of the same word. All of these steps aim to normalize the input and highlight the most informative parts of the text. For example, research surveys emphasize that tokenization and stop-word elimination are important preclassification steps.[9]

After tokenization, vectorization converts text into numerical feature vectors. The simplest approach is bag-of-words (BOW): each unique word in the vocabulary becomes a feature, and a document is represented by the counts (or binary presence) of each word[10] A more refined version is TF-IDF, which weights words by their frequency in the document tempered by how common the word is in the corpus.[9] Both BOW and TF-IDF yield high-dimensional sparse vectors that traditional classifiers can use. Feature selection or dimensionality reduction (e.g. chi-square selection, mutual information, PCA) are sometimes applied to focus on the most relevant words.

In deep learning approaches, word embeddings replace sparse features. Pre-trained embedding models (such as Word2Vec, GloVe) provide dense vector representations of words that capture semantic similarity. Alternatively, embeddings can be learned during model training. As Kowsari et al. note, in an RNN text classifier “each input word is represented by a specific vector using word embedding technology. Then the embedding word vectors are fed into RNN cells one by one.”[21] Modern systems often use contextualized embeddings from pre-trained transformers (BERT, ELMo, GPT) as features. In fact, embeddings from large language models themselves can serve as the input “features” for classification. For instance, one may encode a sentence with BERT’s final layer vectors and then feed those into a simple classifier. This approach leverages massive pre-training to provide rich initial features. Other feature-engineering techniques include n-grams (features representing short phrases rather than single words), part-of-speech tags, and document-level features (length, metadata). Some systems use topic modeling (e.g. LDA features) or dependency-based features for specialized domains. In practice, the choice of preprocessing and features depends on the model: traditional methods rely heavily on manual feature design (n-grams, TF-IDF, etc.), whereas deep and transformer models largely automate feature learning. Nevertheless, even deep models often benefit from basic normalization (e.g. lowercasing, tokenization) and pretrained embeddings.[9][10]

6. Common Datasets and Benchmarks

Text classification research relies on several standard datasets and benchmarks to evaluate models. These include datasets from various domains (news, reviews, social media) and classification types (sentiment, topic, genre). Some widely used examples are:

IMDB Movie Reviews: A binary sentiment dataset with 50,000 labeled movie reviews (25k for training, 25k for testing) from IMDB.[22] Reviews are labeled positive or negative, with an equal number of examples in each class. This dataset is a standard benchmark for sentiment analysis. (Statistics: 50k documents, average length ~294 words.[22])

20 Newsgroups: A collection of about 20,000 newsgroup posts across 20 different topic categories (such as science, politics, sports). Each document is labeled with one of 20 topic classes. It is a classic benchmark for multi-class topic classification. The original dataset has 18,846 documents in total,[23] commonly used with the "bydate" split.

AG's News: A news classification dataset with 120,000 training and 7,600 test samples, categorized into 4 topic classes (World, Sports, Business, Sci/Tech). It is often used for text classification evaluation. (According to Li et al., AG News has 4 classes, ~127,600 examples in total.[23])

Other notable benchmarks include Stanford Sentiment Treebank (SST) for fine-grained sentiment (movie sentences, both 5-class and 2-class versions), Reuters-21578 (newswire articles with multiple topic categories), Yelp Review data for sentiment (positive/negative and 5-star variants), Amazon Review subsets, and DBpedia (Wikipedia abstracts categorized by type). Table 2 of Li et al. (2021) summarizes many of these: e.g., 20NG (20 classes, 18,846 samples), AG News (4 classes, 127,600 samples),[23] SST (2 or 5 classes, ~10k samples),[24] etc.

In addition to these data sets, more recent benchmarks include multi-label datasets (such as EUR-Lex for legal text with many overlapping categories) and cross-domain corpora. The NLP&CC 2013 and tweet sentiment datasets (Twitter, YouTube comments) test models on social media data. Researchers also often perform experiments on custom or domain-specific data (e.g. clinical text, customer reviews). Nonetheless, IMDB, 20NG, AG News, and similar well-known datasets remain go-to benchmarks for demonstrating text classification performance.[22][23]

7. Evaluation Metrics

Evaluating text classification models requires appropriate metrics. The most basic metric is accuracy, defined as the proportion of documents for which the predicted label matches the true label. While accuracy is intuitive, it can be misleading on imbalanced datasets (where some classes dominate).[25] Therefore, practitioners also use other measures:

Precision and Recall: For each class (often focusing on a positive class), precision = TP/(TP+FP) measures the correctness of positive predictions, while recall = TP/(TP+FN) measures the coverage of actual positives. These are important when the cost of false positives vs false negatives differs.

F1-Score: The harmonic mean of precision and recall, F1=2·(precision·recall)/(precision+recall). F1 is especially useful in imbalanced settings, as it balances precision and recall. In multi-class problems, one can compute a macro-averaged F1 (averaging F1 across classes) or micro-averaged F1 (computing metrics globally).

ROC-AUC: For binary classification, the area under the receiver operating characteristic curve (ROC-AUC) is often reported. ROC-AUC measures the trade-off between true positive rate and false positive rate across decision thresholds. It is threshold-independent and useful when classes are imbalanced, though it is less commonly used for multi-class classification.

Other Metrics: Some works also report precision@K or NDCG@K (for ranked outputs), average precision, or log-loss. For hierarchical or multi-label classification, specialized metrics like hierarchical F1 or Hamming loss may be used. However, for standard single-label classification, accuracy and F1 are the most common metrics.[26] Li et al. summarize that most text classification papers report accuracy and often F1 as well.[25] For example, RoBERTa and ALBERT papers show both F1 and accuracy metrics on benchmarks.[27]

In summary, accuracy and F1 are the default metrics for reporting classification performance. When datasets are imbalanced (common in text tasks), F1 (or precision/recall) provides more insight than raw accuracy. Researchers should always choose metrics aligned with the application's priorities (e.g., high recall for medical alert detection, high precision for spam filtering). As one survey notes, a critical part of model evaluation is measuring and comparing performance via these metrics.[26]

8. **Real-World Applications**

Text classification powers many real-world systems. Notable application areas include:
Sentiment Analysis: Automatically detecting sentiment or opinion polarity in text (e.g. positive/negative reviews). Sentiment analysis is essentially a text classification task where the categories are sentiments (positive, negative, neutral).[2] It is widely used to understand consumer opinions, monitor social media, or gauge public mood. For example, classifying movie or product reviews as positive or negative is a canonical sentiment analysis application.[28][2] Social media companies and businesses use sentiment classification to filter content or summarize customer feedback.

Spam and Email Filtering: Classifying emails or messages as spam (junk) or not spam. This was one of the earliest successes of text classification. Stanford's IR textbook cites spam detection as a prototypical use case.[29] An automated spam filter labels incoming email using a text classifier, protecting users from unwanted messages. The same idea extends to detecting phishing, harmful content, or email triage into priority folders.

Topic/Content Classification: Assigning categories or topics to documents. News agencies and libraries classify articles by subject (politics, sports, technology, etc.), which is often phrased as topic classification. For example, organizing news articles into subject-based categories enables topic-specific search (e.g. a "vertical search" engine indexed only on computer science departments.[30]). Commercial applications include categorizing customer service tickets, organizing legal or medical documents, or tagging articles for recommendation systems. More generally, any task that involves classifying the theme or genre of a document (e.g. classifying research papers by field, tweets by topic) falls under text classification.

Other real-world tasks include language identification (classifying which language a text is in), authorship attribution (who wrote a text), intent classification (in dialog systems), toxic content detection (classify text as abusive or not), and many domain-specific uses. In e-commerce, text classifiers predict product categories from descriptions; in healthcare, they categorize clinical notes by diagnosis. The survey of approaches highlights that sentiment analysis, spam detection, and topic categorization remain the most common applications.[2][29] These applications have driven advances in classification methods, as practical systems demand both high accuracy and scalability.

9. Challenges and Limitations

Despite great progress, text classification faces several persistent challenges:
Data Imbalance: Many real-world datasets have imbalanced classes (some labels are much more frequent than others). For instance, in fraud detection or rare event monitoring, the "positive" class may be tiny. Imbalance can cause models to be biased toward the majority class, inflating accuracy but missing rare cases. Handling imbalanced datasets requires special techniques (resampling, class-weighted loss, anomaly detection framing).[25] Li et al. highlight that dealing with imbalanced datasets is a key difficulty in text categorization.[25]

Interpretability: Modern text classifiers (especially deep and transformer models) are often "black boxes." It can be difficult to understand why a model made a certain classification. This lack of transparency is problematic in sensitive domains (finance, law, healthcare) and can erode user trust. As one explainability review notes, deep NLP models "typically lack transparency" and balancing high accuracy with interpretability remains an unresolved issue.[31] Regulatory pressures (e.g. GDPR's "right to explanation") further demand that models be interpretable.[32] Thus, providing meaningful explanations for classifications (e.g. highlighting key input phrases) is an active area of research, but remains challenging.
Domain Shift and Adaptation: Models trained on one domain of text may perform poorly on another. For example, a sentiment classifier trained on movie reviews might fail on product reviews or tweets, due to differences in language style and vocabulary. Adapting models across domains (or languages) without large amounts of labeled data is difficult. Domain-specific jargon or formatting (e.g. legal documents vs. social media) can confound classifiers. Solving this requires either collecting new labeled data per domain or using transfer learning methods. Even with transfer (fine-tuning on new data), domain adaptation is nontrivial and remains an open research question in text classification.

Short and Noisy Text: Social media texts and user-generated content are often very short, misspelled, or full of slang and emojis. Classifying such noisy, underinformative inputs is harder than clean, well-edited text. Simple BOW models may struggle when texts are short (e.g. tweets), and language variation (dialects, code-switching) adds complexity. Preprocessing (like normalization) helps, but robust performance on "wild" text continues to be a limitation for many systems.
Interpretation of Multilabel and Hierarchical Classes: Some tasks involve multilabel classification (documents belong to multiple classes) or hierarchical taxonomy of classes. Handling these requires more complex modeling (e.g. threshold tuning or using label hierarchies) and complicates evaluation. Many classification techniques are inherently single-label and need extensions for multilabel settings.

Computational Resources: Training large models (like deep neural networks or BERT) can be resource-intensive. Organizations with limited compute may find it hard to train or even fine-tune big transformer models. This raises issues of scalability and environmental cost.

In summary, the main limitations of current text classification approaches involve data issues (imbalance, domain mismatch), model explainability, and scalability constraints. Addressing these challenges is essential for deploying classification in real-world settings.[25][31]

10. Recent Trends and Future Research Directions

Text classification research is rapidly evolving. Several key trends and future directions include:

Pretrained Large Language Models and Few-Shot Learning: The dominance of pre-trained transformers has led to a shift toward models that can learn from very few examples. Researchers are exploring zero-shot and few-shot classification via prompting large language models like GPT-3/4. For instance, recent work shows that GPT-based models, given a handful of examples in a prompt, can achieve competitive performance on classification tasks.[20] This few-shot paradigm reduces the need for large, labeled datasets. It is likely that future systems will blend fine-tuning and prompt-based learning, and that few-shot learning techniques (meta-learning, prompt engineering, etc.) will become standard, especially for low-resource scenarios.

Multilingual and Cross-Lingual Models: With globalization, there is growing interest in models that work across languages. Multilingual BERT (mBERT) and XLM-RoBERTa are pretrained on many languages and can be fine-tuned for classification in a target language, or even for zero-shot cross-lingual transfer. As noted in surveys, dealing with multilingual texts is already recognized as an important challenge.[25] Future research will push in making models more robust across languages and dialects, and in creating better multilingual benchmarks.

Explainable and Interpretable Classification: As discussed, there is strong impetus to make classifiers more explainable. Techniques such as attention visualization, feature attribution (LIME, SHAP), example-based explanations, and context-aware explanation networks are emerging. For example, recent work proposes context-aware explanation modules that tailor explanations for different audiences.[31][33]. We expect explainability to remain a major trend: models will likely incorporate built-in interpretability (e.g. simpler hybrid models) or advanced post-hoc explanation techniques to justify their predictions.

Domain Adaptation and Robustness: Models that generalize across domains or adapt quickly to new domains are another research focus. Approaches may include unsupervised domain adaptation, adversarial training, or continual learning methods. There is also interest in improving robustness to noise, adversarial attacks, and distribution shifts (for instance, classifying social media text over time as language evolves).

Few-Shot and Meta-Learning: Beyond prompt-based few-shot, meta-learning and transfer learning approaches (like Siamese networks or prototypical networks) are being applied to text classification. These techniques aim to rapidly adapt to new classes or tasks with minimal data.

Ethical and Fair Classification: Ensuring fairness and avoiding biased classifications is increasingly important. Research into detecting and mitigating biases (e.g. gender or racial bias in text classifiers) is growing. Relatedly, privacy-preserving classification (e.g. federated learning for text) is an emerging area.

Lightweight and Efficient Models: There is also a push toward smaller, faster models (distillation, quantization) that can run on devices or in low-resource settings while retaining accuracy.

In summary, future research in text classification is moving toward more flexible, data-efficient, and transparent methods. Large pre-trained transformers with few-shot learning are currently "state of the art," but their reliance on massive data is prompting work on alternative paradigms. Enhancing multilingual capability and model interpretability are seen as high priorities.[25][32] We expect to see continued innovation in leveraging unlabeled data, better understanding of model behavior, and new benchmarks that reflect real-world diversity of languages and domains.

References:

Aggarwal, C. C., & Zhai, C. (2012). *Mining text data*. Springer. https://doi.org/10.1007/978-1-4614-3223-4

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Galke, L., Scherp, A., & Stumme, G. (2017). A survey on automated document classification for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37–38, 83–95. https://doi.org/10.1016/j.websem.2015.09.002

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 427–431.

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4), 150. https://doi.org/10.3390/info10040150

Li, Y., Yang, T., Zhang, Y., & Wang, X. (2021). A survey of deep learning-based text classification. *IEEE Access*, 9, 14653–14675. https://doi.org/10.1109/ACCESS.2021.3056780

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., … Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*. https://arxiv.org/abs/1907.11692

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. https://arxiv.org/abs/1301.3781

Nguyen, T. H., & Grishman, R. (2015). Relation extraction: Perspective from convolutional neural networks. *Proceedings of NAACL-HLT 2015*, 39–48.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 1–47. https://doi.org/10.1145/505282.505283

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of EMNLP 2013*, 1631–1642.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 28, 649–657.

Zhao, W. X., Zhou, J., Li, Z., Wang, W., & Wen, J. R. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*. https://arxiv.org/abs/2303.18223