Explainable and Transferrable Text Categorization

Tobias Eljasik-Swoboda[1](✉), Felix Engel[2], and Matthias Hemmje[2]

[1] Faculty of Mathematics and Computer Science, University of Hagen, Hagen, Germany
Tobias.Swoboda@fernuni-hagen.de
[2] FTK e.V. Forschungsinstitut für Telekommunikation und Kooperation, Dortmund, Germany
{fengel,mhemmje}@ftk.de

Abstract. Automated argument stance (pro/contra) detection is a challenging text categorization problem, especially if said arguments are to be detected for new topics. In previous research, we designed and evaluated an explainable machine learning based classifier. It was capable to achieve 96% F1 for argument stance recognition within the same topic and 60% F1 for previously unseen topics, which informed our hypothesis, that there are two sets of features in argument stance recognition: General features and topic specific features. An advantage of the described system is its quick transferability to new problems. Besides providing further details about the developed C3 TFIDF-SVM classifier, we investigate the classifiers effectiveness for different text categorization problems spanning two natural languages. Besides the quick transferability, the generation of human readable explanations about why specific results were achieved is a key feature of the described approach. We further investigate the generated explanation understandability and conduct a survey about how understandable the classifier's explanations are.

Keywords: Argument stance detection · Explainability · Machine learning · Trainer-athlete pattern · Ontology creation · Understandability support vector machines · Text analytics · Architectural concepts

1 Introduction

The goal of the RecomRatio project is to implement an information system for supporting medical professionals. This support comes in the form of recommendations for treatment options in combination with rational arguments why specific treatments are suggested. In the envisioned system, the recommendation is going to be based on an argument ontology containing entire pro and contra arguments for given topics in medicine. These arguments are sourced from concrete clinical studies and other publicly available data sets like PubMed [1]. The ability to explain why certain recommendations were generated is important due to two reasons: Firstly, patients and medical practitioners have more confidence in a recommendation if they can inspect the reasons for this argument. Secondly, the European Union's (EU) General Data Protection Regulation

S. Hammoudi et al. (Eds.): DATA 2019, CCIS 1255, pp. 1–22, 2020.
https://doi.org/10.1007/978-3-030-54595-6_1

(GDPR) contains a right to explanation [2]. This piece of legislature grants every EU citizen the right to demand explanations for the results of machine learning (ML) and artificial intelligence (AI) systems if they are impacted. Up until recently, this has not been an aspect of ML and AI research [3]. At the time of writing this article, PubMed contained over 30,000,000 medical abstracts and links to the actual papers and studies [1]. Manually assessing this much data and modeling its contents into ontologies is not feasible. To overcome this bottleneck Argument Mining (AM) is employed.

Research in Argument Mining aims to automate the detection of arguments in large amounts of text. It combines a broad set of computer science sub disciplines such as Natural Language Processing (NLP), Artificial Intelligence (AI) and Computational Linguistics [4]. A specific challenge within Argument Mining is the detection of an argument's stance. This means whether it is pro or contra to a specific topic. At the Semeval16 conference, multiple stance detection approaches have been evaluated [5]. Detecting the stance of arguments about previously unseen topics is more challenging than detecting them in known topics.

This paper is the extended version of a conference paper in which we described an explainable, machine learning based classifier, called C3 TFIDF-SVM, which was evaluated using two argument stance detection datasets [6]. The C3 TFIDF-SVM classifier is intended as part of the argument mining environment used to create the RecomRatio argument ontology. The developed classifier was capable to achieve .96 F1 within the same topic and $>.6$ F1 for different topics. This informed our following hypothesis:

There are two sets of terms that serve as argument stance features:

1. *The set of general argument stance feature G.*
2. *The set of topic specific argument stance features F(t).*
   If one has G and F(t) for topic t along with a machine learned model for the combination of these features, high effectiveness, explainable classification can be achieved. If one works with another topic, F(t) becomes noise decreasing overall effectiveness.

This article provides further details about the C3 TFIDF-SVM and analyzes our hypothesis. C3 TFIDF-SVM has the additional advantage of being quickly transferable to other TC problems. Therefore we tested it on four different TC problems across two natural languages and evaluated its performance. Additionally, we conducted a survey about how understandable the generated explanations are.

The remainder of this article is structured as follows: Section two describes the state of the art and technology relevant for the creation of C3 TFIDF-SVM. Section 3 provides details about the underlying model used for C3 TFIDF-SVM. Section 4 details its implementation. Section 5 contains an effectiveness evaluation across multiple problems while Sect. 6 contains our survey results about the generated explanations' understandability. Section 7 finishes with conclusions about our research.

2 State of the Art

2.1 Argument Mining

As previously explained, Argument Mining or Argumentation Mining aims to automate the detection of arguments in large amounts of text. Besides the detection of arguments

related to specific topics, detection of the argument stance is a specific challenge. In 2016, the association for computational linguistics created the *detecting stance in tweets* challenge. The results were published in the proceedings of SemEval-2016 [5]. For this challenge, 4,870 English tweets for stance towards six commonly known topics in the United States were annotated with *favor* and *against*. These topics were *Atheism*, *Climate Change is a Real Concern*, *Feminist Movement*, *Hillary Clinton* and *Legalization of Abortion*. Here, 70% of the annotated data was provided for training while the remaining 30% were reserved for testing. 16 Teams submitted classifiers for this task where the highest F1 result was 67.82%. It was achieved by employing two recurrent neural network classifiers in concert. This however was below the baseline classifier created by the challenge's organizers which achieved 68.98% F1 by using a linear kernel Support Vector Machine (SVM) (see Sect. 2.2) classifier per topic that used word n-grams (1-, 2-, and 3-gram) and character n-grams (2-, 3-, 4-, and 5-grams) as features [7]. This was subsequently tuned using 5-fold cross-validation on the available training data. To the best of our knowledge, no further feature selection took place, providing the SVM with a high dimensional problem. It is noteworthy, that this comparatively simple baseline model outperformed many more advanced classifiers using deep learning or word embeddings (see Sect. 2.2).

For the second task of the challenge, the new topic *Donald Trump* was introduced. Here, no labeled training data was provided and the classifiers trained on the previous topics were tasked with classifying tweets towards this new topic. The best participant achieved 56.28% F1. The baseline system was used in two ways: Firstly, a majority vote of all individual SVMs regarding the stance of the new topic was performed. Secondly, one SVM was trained with training samples across all different topics. The Majority Baseline classifier achieved 29.72% overall F1. Interestingly, it hat an F1 of 59.44% for identifying the *against* sentiment while that to identify the *favor* sentiment is 0.

In their work Stab et al. created the UKP Sentential Argument Mining Corpus (UKP) including over 25,000 instances over eight topics by querying Google regarding these topics and having crowd workers annotate Google's preview texts [8]. The UKP corpus covers the topics *abortion*, *cloning*, *death penalty*, *gun control*, *marijuana legalization*, *minimum wage*, *nuclear energy* and *school uniforms*. Using a LSTM based classifier, Stab et al. were able to achieve F1 of 66.62% within the same topic. Some interesting experiments with the training set were performed. The classifier was evaluated with a single topic. It was trained on a mix of arguments containing different percentages of target topic samples. 20% of target topic data already have strong positive effects on recall (see Sect. 2.3).

Another task in this context is same side stance classification. Here, two arguments are used as input and a classifier has to determine if both have the same or different stances. Within the same topic, F1 values of 77% are state of the art. In different topics, F1 values of 73% were achieved [9].

There are multiple other works in the field of argument mining which cannot feasibly be covered in a single article. The illustrated works however show the following findings: Firstly, argument stance recognition is hard. State of the art systems struggle to achieve $>70\%$ F1. Secondly, argument stance recognition is even more difficult when a classifier is created using one set of topics but is subsequently applied to another topic.

Thirdly, even though SVMs have been around for a long time and linear kernels are the simplest implementation of SVM; they are surprisingly highly effective compared to more sophisticated classifiers.

2.2 Text Categorization Approaches

The formal definition of Text Categorization (TC) is that a classifier Φ: $(D, C) \rightarrow \{0, 1\}$ approximates a target function Φ': $(D, C) \rightarrow \{0, 1\}$ as closely as possible [10]. Here the set D contains relevant documents and C the categories they are assigned to. For the argument stance recognition problem, D consists out of individual arguments and C of *pro* and *contra*. Of course there are a multitude of different NLP problems that can be modelled as TC problem. Among these problems are *intent detection* for chatbots or *hate speech detection* for social media applications.

The set of available assignments $\Omega = \{\Phi', D, C\}$is also referred to as *initial corpus.* For research purposes Ω is oftentimes split into training sets *TS* and evaluation sets *ES* so that $TS \cap ES = \emptyset$. While *TS* is used to create Φ, *ES* is used to evaluate its effectiveness using different metrics described in Sect. 2.3. If no predetermined split between *TS* and *ES* is available, one can also use the *n-fold cross-validation* process. Here, Ω is split in n subsets of about equal size. In the next step, n different classifiers are created, each of which are created with a different combination of $(n - 1)$ subsets so that the remaining subset can be used as *ES*. One can of course perform n-fold cross-validation on a predetermined training set and then apply the best performing model to the actual evaluation set of a specific task. For practical applications, the evaluation set is the real life productive application of the classifier in an application.

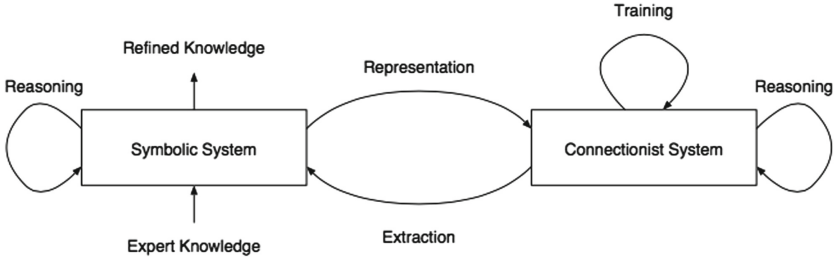

Fig. 1. Illustration of nerual-symbolic integration [11].

There are two fundamental approaches of how to implement TC: One can either used fixed rules or apply machine learning to the problem [10]. The combination of both is referred to as neuro-symbolic integration (see Fig. 1) [11]. For rule based TC, experts need to create rules which a machine can use to determine the correct category. To do so, texts are checked for the occurrence of certain terms which can be simple words or more complex regular expressions describing entire sentences [10, 12].

Most machine learning based classification algorithms are defined for numeric input [10]. Therefore machine learning based text categorization occurs in three phases each of which can be implemented using different algorithms or algorithms spanning multiple of these phases:

1. Phase: Feature extraction: Text is transformed into feature vectors comprehensible for machine learning algorithms.
2. Phase: Feature selection. Reduction of the feature vector size to speed up and enhance the effectiveness of machine learning
3. Phase: Machine learning based text categorization.

For practical applications, one has to distinguish between the training of classifiers, or models, that then work on productive data during inference. Popular approaches for the implementation of machine learning are simulated neural networks [13, 14]. Deep learning was originally defined as using cascades of multiple machine learning approaches but has since also become synonymous with artificial neural networks that use more than three layers or neurons between in- and output. These approaches are so popular that they have become synonymous with machine learning and artificial intelligence albeit multiple other highly effective approaches exist [15]. One such alternative are Support Vector Machines (SVMs) [16]. These operate by projecting input data points into high-dimensional spaces which are bisected by a hyperplane. If the points are above the hyperplane they belong to the category, if not they are not, they don't belong to the category. Training the hyperplanes from samples is performed by maximizing the margin between the hyperplanes and the closest vectors. As only the closest vectors are taken into consideration, these are also referred to as support vectors. In order to cope with non-linearly separable datasets, SVMs can work with slack variables that essentially allow for certain vectors to be a little bit on the wrong side of the hyperplane. Additionally, they can work with alternative kernels. In the context of SVMs, kernels are alternative implementations of the dot-product which is required to compute the relative location of a data point related to the hyperplane. Changing its computation allow for curved hyperplanes that essentially project data into higher dimensional spaces so that it can be bisected with a hyperplane there. As Sect. 2.1 has shown, even linear kernel SVMs (where the dot product simply is the dot product) were able to outperform sophisticated neural networks in certain argument mining tasks. A common library for the implementation of Support Vector Machines is LibSVM [17].

With regards to TC, SVMs are phase 3 algorithms. Their performance largely depends on how data points are represented to them. A straight forward approach is to count how often certain terms occur in each text and use these values as feature vector. This is referred to as the Bag of Words (BoW) approach. One can fine-tune it by providing a controlled vocabulary of relevant terms to spot. One however needs to normalize these vectors so that the generated vectors are of equal length. More complex approaches are to use word n-grams (n words occurring after each other) and character n-grams (n single characters occurring after each other) as features. This can quickly lead to highly dimensional text representations which in turn require large amounts of computational resources for training and inference. An alternative are Word Embeddings. This class of unsupervised learning algorithms uses unlabeled text to represent terms with coordinates which encode their relationship to each other based on their offset. For instance, the offset between terms of different gender like *man*, *woman*, *boy*, *girl*, *king*, and *queen* are all similar. Notable implementations are Word2Vec's CBOW, skip-gram, and GloVe [18, 19]. During the training of word embeddings, individual texts can also be represented by a vector. This however is not possible during inference. The problem of

grouping a sequence of word vectors into a single vector representative of the sequence is called Compositional Distributional Semantics (CDS) [20]. Multiple approaches such as the Basic Additive Model or the Word Mover's Distance (WMD) have been proposed [21]. These have been used to implement entirely unsupervised classifiers that work by assessing the distance between categories and texts [22, 23].

Another notable approach to feature extraction is the use of neural network based encoders/decoders. Their aim is to transform terms into numeric vectors and vice versa. Different from word embeddings, these encoders take the ordering and accompanying words into account. This allows it to determine what words like *it* in a sentence refer to [24]. BERT is available as pre-trained model that was created in an unsupervised fashion. The model can subsequently be used in any NLP application. BERT models are huge. The BERT_large model has 345 million individual parameters making up the network. The BERT_base model still has 110 million parameters. Both are highly obscure meaning that their results are next to impossible to explain.

As initially stated, the explainability of generated results is a comparatively novel requirement that has not been in the focus of NLP research until recently. In principle, researchers and programmers that are intimately familiar with their algorithms and have access to their internal state at run time, for instance by accessing log files, can provide explanations why certain results were created. This is however insufficient for practical applications because the internal state usually is not logged and the creators of algorithms are not the organizations operating them and having to explain why certain results were generated. Operating organizations actually require a system that can automatically create explanations for generated results. Lexicon based classifiers, such as *ReLexNet*, are a class of such easily explainable classifiers [3]. These comparatively simple constructs function by creating an association matrix between individual terms and categories. The likelihood of a document to belong to a specific category is based on how many terms of high association with this category occur within the document. These terms can be influenced by modifier and negator terms. Negators multiply the term's association with -1. Strength modifiers like *little* can reduce or boost an association value, for example by 0.3. Explanations can be generated by stating which terms were found within the documents. A drawback of lexicon based classifiers are, that they require manually created lexica of at least modifier terms. Additionally, they can have issues with linear separability, as two words individually might indicate a certain category but in combination would contra indicate it. This is difficult to model using only a lexicon.

As described in this section, there are many options and considerations when implementing a TC application like argument stance recognition. The Cloud Classifier Committee (C3) was developed to make TC easily available to any application. To make TC as simple as accessing an external database, C3 was implemented as modern and flexible microservice-oriented architecture [6, 12, 25]. C3 provides TC capability through a simple REST/JSON interface. Notably, this interface caters to generate explanations for created categorizations. To do so, the trainer/athlete pattern for microservice oriented machine learning is used (see Fig. 2).

With the trainer/athlete pattern, a trainer service computes a model that can be used by any athlete service. As applied inference can be time critical for certain applications, the athlete services are stateless after having been provided with a model. This way one

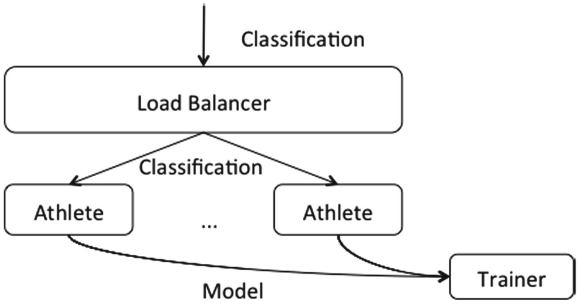

Fig. 2. The trainer/athlete pattern for microservice-oriented machine learning [6].

can easily scale-out the application as required. As TC uses common API endpoints, one can combine different C3 microservices into committees of automated classifiers.

2.3 Feature Assessment and Evaluation Metrics

Feature rating, assessment, selection and extraction as used in this publication build on concepts from the Information Retrieval (IR) community. To evaluate the ability of an IR system in a specific task the two basic metrics *precision* and *recall* are used. Precision is defined as the proportion between the intersection of relevant and retrieved documents to retrieved documents. One could say it is the percentage of correctly retrieved documents. In contrast to precision, recall is defined as the proportion of the intersection between relevant documents and retrieved documents to relevant documents. In other words the percentage of how many documents were retrieved to how many documents should have been retrieved. Both measures are excellent to gauge how correct a TC system operates for a specific category. The F1 score is the harmonic mean between precision and recall. One can for instance have perfect recall but poor precision by assigning every document to one category. A low F1 value shows such shortcomings. TC is oftentimes performed for more than one category. There are two different methods for averaging result scores: In Microaveraging, individual true positives, false negatives and false positives of each category are summed up and global precision, recall and F1 scores are calculated. In Macroaveraging, the results of all individual categories are averaged [10].

The *Term Frequency Inverse Document Frequency* is a statistic value used to indicate how representative a term is for a specific document (see Eq. 1).

$$tfidf\left(t_k,\, d_j\right) \;=\; \#\left(t_k,\, d_j\right) \,*\, \log\left(\frac{|TS|}{\#TS(t_k)}\right) \tag{1}$$

TFIDF is computed by counting how often a specific terms t_k occurs within document d_j ($\#(t_k, d_j)$) and multiplying this value with the logarithm of the quotient between the training set size ($|TS|$) and the amount of documents in the training set that contains term t_k. This formula models two intuitions: Firstly, if a term occurs oftentimes within the same document, it is very representative for the document. Secondly, if a term occurs in many documents, it cannot be representative for any document. If a term occurs in all documents, it has a TFIDF value of 0.

3 Model

The goal for our model is two-fold. Firstly, we aim to create an explainable classifier for argument stance recognition. Secondly, we want to identify relevant features for the identification of topic specific arguments.

If a classifier has sets G and $F(t)$ of topic specific and general features (see Sect. 1), it could generate explanations for categorization decisions based on found features within the text. Explanations can be stored in sentences like:

> *"The argument is considered contra, because it contains multiple occurrences of the terms tragedy, leaks, soldiers and blood. These terms have been detected as contra indicators in the given data set containing 27,538 arguments about the topic policy"* [6].

Having multiple topic specific data sets enables the creation of an argument stance feature ontology by identifying sets of relevant features and comparing these features with each other. As shown in Fig. 3, a concept can be an argument stance feature which can either be topic specific or general. In any case, it was extracted from a specific corpus of documents which in turn is specific to a topic.

To achieve our goals, we require a model that can automatically detect relevant features for argument stance recognition. To do so, we implemented the C3 trainer/athlete pattern. When creating a model, the trainer is provided with an initial corpus of labeled arguments. Based on this corpus, the TFIDF value for every term t_k is computed. To do so, the word occurrences in all documents are counted. No filters like stop word lists or predefined n-gram lists are applied as the goal overreaching concept of C3 is to make TC as accessible as possible by requiring no additional resources but the training set. Stop word lists are almost superfluous as stop words have low TFIDF values as they occur in almost all documents.

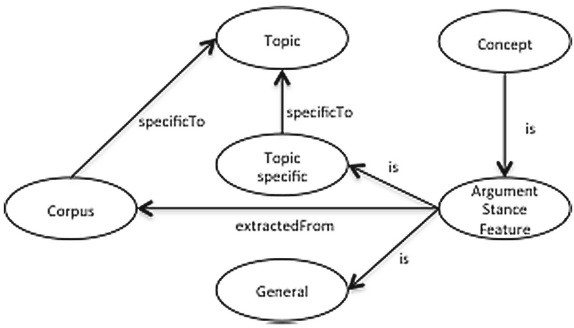

Fig. 3. Structure of a topic specific argument stance feature ontology [6].

After computing the TFIDF values, relevant feature terms must be identified. This is done by identifying the h highest TFIDF values for documents belonging to each category. We chose 20 as default value for this configurable parameter. This means that h relevant terms per category are added to an automatically created controlled vocabulary.

In the next step, this list of relevant features is augmented by the single highest TFIDF term of every available document if it is not already within this list. This approach creates a list of word 1-grams that serve as features for the TC task the list was created for.

We chose to use LibSVM with a linear kernel as machine learning based classification approach because of multiple reasons: Firstly, linear kernel SVMs have been reported to perform better than many more sophisticated approaches in argument mining (see Sect. 2.1). Secondly, SVMs are relatively easy to explain: If a feature vector is above the hyperplane, the document belongs to the category. Thirdly, SVMs require comparatively little computational resources and their stored models are highly portable as the hyperplane can easily be stored and transferred to other machines.

LibSVM requires its feature vectors to be normalized per dimension. This means, that in all feature vectors of a corpus, the scalars representing an individual term (i.e. *tragedy*) must have a combined length of 1. LibSVM also requires sparse vectors consisting out of arrays containing topics of the dimension and value for this dimension. To create this representation, the dimension is specified by the term's id within the controlled vocabulary. The value is the normalized TFIDF value. Formula 2 provides the necessary dimensionally normalized value.

$$ntfidf\left(t_k,\, d_j\right) = \frac{\#\left(t_k,\, d_j\right) * \log\left(\frac{|TS|}{\#TS(t_k)}\right)}{\sqrt{\sum_{i=1}^{|D|}\left(tfidf\left(t_k,\, d_i\right)\right)^2}} \tag{2}$$

During training, this is easily computed, as the original TFIDF matrix for selecting relevant features is still available. In order to use this scheme during inference in an athlete service, one must take into account that every document from which features are extracted is actually the first one after |TS|. This changes part of the original TFIDF equation from $\frac{|TS|}{\#TS(t_k)}$ to $\frac{|TS|+1}{\#TS(t_k)+1}$. As the trainer service does not contain the original TFIDF matrix, the transferred model must contain $\sqrt{\sum_{i=1}^{|D|}\left(\#(t_k,\, d_i) * \log\left(\frac{|TS|+1}{\#TS(t_k)+1}\right)\right)^2}$ for each term t_k. This way, inference in an athlete service can use formula 3 because the denominator, $|TS|$ and $\#TS(t_k)$ are stored in the model:

$$itfidf\left(t_k,\, d_j\right) = \frac{\#\left(t_k,\, d_j\right) * \log\left(\frac{|TS|+1}{\#TS(t_k)+1}\right)}{\sqrt{\sum_{i=1}^{|D|}\left(\#(t_k,\, d_i) * \log\left(\frac{|TS|+1}{\#TS(t_k)+1}\right)\right)^2}} \tag{3}$$

For training, n-fold cross-validation on the initial corpus is performed. This creates n models each of which is evaluated with a different evaluation set. Afterwards, the trainer selects the model of highest effectiveness. Which effectiveness measure is used can be selected. The default value is microaverage F1 (see Sect. 2.3). The best performing TFIDF model as well as the controlled vocabulary and the afore-mentioned relevant values of formula 3. This creates a compact model that is freely transferrable from trainer to athlete services without transferring the training set. As it contains a list of relevant terms to use as features, these terms can be used to generate explanations as well as to create the argument feature ontology shown in Fig. 3.

4 Implementation

Based on the utilized technologies, we refer to our classifier as C3 TFIDF-SVM. The trainer/athlete pattern and all required interfaces are implemented as REST/JSON endpoints. These endpoints provide Create, Read, Update and Delete (CRUD) operations which can be triggered using the http POST, GET, PUT and DELETE verbs. These endpoints are application independent and cater for documents, categories, category relationships (hierarchical categories are possible), manual assignments (the target function), categorizations (the athlete's results) created models and their effectiveness evaluations for the trainer and actively used model for the athlete. Additionally, a metadata endpoint provides information about the service. Besides these endpoints, C3 services also include a web GUI which controls the service using the endpoints and is hosted on it. The categorization objects link documents with their categories and additionally contains a human readable explanation. An advantage of the JSON objects are the relatively simple human readability which allows us to inspect which terms were selected to be used as features.

We implemented the services in Java using the Dropwizard framework [26]. This approach allows us to package the classifier as fat jar file which can be run on any device that support Java and has a network interface. There is only one additional YML file required which contains parameters such as the network port the service is listening on. In terms of machine learning libraries, only LibSVM was used. We implemented the described feature extraction and selection scheme directly. In addition to this jar packaging, we additionally packaged the services as docker containers in order to minimize deployment time on arbitrary environments [27].

Another part of C3 is an API library that can be used in java in order to access the C3 endpoints. This is however completely optional, as all experiments can also be performed via the web-GUI. Using the library is however much more convenient for thousands of arguments to be uploaded.

As shown in Fig. 4, the TFIDF-SVM service was executed on a Linux server provided by the university that was accessed by a personal computer similar to how a piece of software accesses a central database. A Microsoft Azure Container Registry was used to store the container which necessitated the installation of the Microsoft *az* command line utility on the server. This setup underlines the flexibility of C3 as Apple, Microsoft and Linux systems were easily combined.

The TFIDF-SVM trainer automatically performs n-fold cross-validation and provides effectiveness results obtained during the training of a model. If the created model is to be evaluated with previously unseen arguments, an athlete service is provided with the created model. Instead of implementing an additional evaluation component, the C3 committee service is used. This service forms a committee of multiple athletes. As combination function, it measures the effectiveness of utilized athletes with all known documents. The results are subsequently weighted according to their effectiveness. If only a single athlete is provided, the C3 committee only measures the effectiveness of this single athlete which is then stored for inspection.

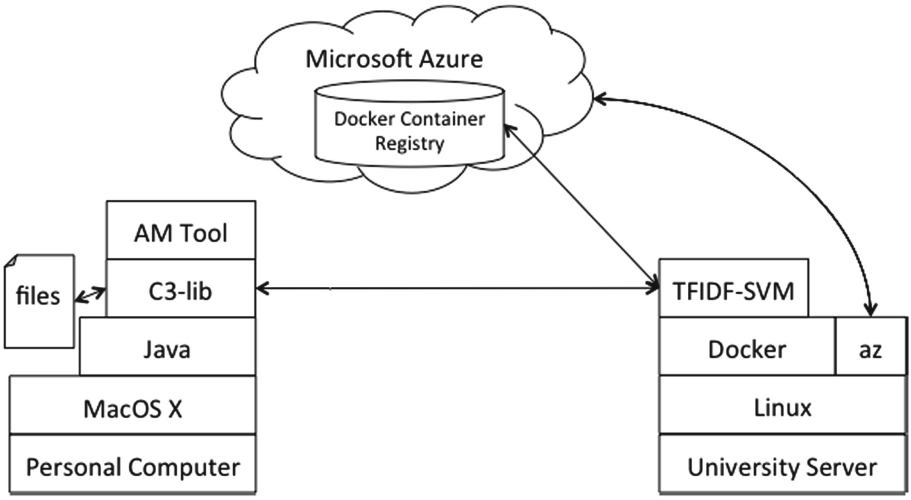

Fig. 4. Flexible setup used for experimentation.

5 Effectiveness Evaluation

5.1 General and Topic Specific Features for Argument Stance Recognition

For our original experiments in the domain of argument stance recognition, we used two different corpora of labelled arguments [6]. The first is the *annotated corpus of argumentative microtexts* (ACAM) [28]. This corpus contains 25,351 argument pairs about 795 different topics. The argument was initially intended for the argument stance same side classification task. As both arguments of every pair are individually labeled, they can be individually analyzed. This corpus is actually an amalgam of two separate corpora. The first was collected during an experiment with 23 participants discussing controversial topics in German. Their arguments were then professionally translated to English. Other argument pairs were written by Andreas Peldszus and are mainly used for teaching activities. This corpus is interesting because of the large amount of topics it covers. To further evaluate C3 we used Stab et al.'s UKP Sentential Argument Mining Corpus introduced in Sect. 2.1.

In our first experiment, we provided C3 TFIDF-SVM with all arguments from the first five topics of ACAM. These were wildly mixed, ranging from the statement that the United States policy on illegal immigration should focus on attrition through enforcement rather than amnesty to a discussion about record stores vs. internet-bought, downloaded music collections. During n-fold cross validation, the trainer service achieved 96% F1. The effectiveness to detect contra arguments was especially high with 100%. Upon inspection of the generated model, it contained terms like *violence*, *national*, or *supporters* for immigration policy or *cd*, *record*, and *collection* for the musical discussion. The second experiment was to work with larger sets of topics. We used the first 10, 20 and 40 topics of ACAM to create models in n-fold cross validation. At this point, a drawback of the TFIDF based feature extraction approach becomes apparent: The amount of features grows only slightly sub-linearly to the amount of arguments as, especially in the

beginning the most representative term per document is not yet part of the automatically created controlled vocabulary. This means that the problem size for the SVM does grow almost quadratically as the dimensions as well as the amount of feature vectors grows. This drastically increases the time requirements to compute models for larger datasets and made experiments with larger datasets unfeasible on our available hardware. The results of these experiments are shown in Table 1.

Table 1. C3 TFIDF-SVM results for ACAM topic mixes using 3-fold cross validation [6].

Topic	1 to 10	1 to 20	1 to 40
Arguments	202	449	813
Terms	84	157	260
F1	88%	89%	85%

To evaluate whether the generated models generalize well, we used the model computed from the first 40 topics of ACAM to initialize an athlete service and evaluate it using a C3 committee service. The results listed in Table 2 indicate how well a model created from 5% of all available topics performs for the remaining topics.

Table 2. C3 TFIDF-SVM results for ACAM topic mixes [6].

Topic	41 to 120	41–200	41–795 (all topics)
Microaverage F1	60%	57%	57%
Microaverage precision	60%	57%	57%
Microaverage recall	60%	57%	57%
Pro F1	23%	21%	21%
Pro precision	17%	16%	15%
Pro recall	35%	34%	35%
Contra F1	73%	70%	71%
Contra precision	83%	82%	83%
Contra recall	64%	62%	62%

Similar to the previous experiments, the effectiveness to identify contra arguments is much better than that to identify pro arguments. It is noteworthy, that C3 TFIDF-SVM doesn't have the information, that both categories are mutually exclusive. Knowing the better performance of the contra category, this can easily be leveraged to increase the overall effectiveness to $>70\%$ F1. For example, only considering something a pro argument if it is identified as pro argument and simultaneously not identified as contra argument will boost the pro-category's precision to at least that of contra-category ($>83\%$ instead of $>15\%$). Creating such informed ruled based on previous performance can be

regarded as neural-symbolic integration even though TFIDF-SVM works with non-neural network based machine learning.

In all previous experiments, 3-fold cross validation was used. For another round of experimentation, we use 10-fold cross validation. As shown in Table 3, switching to 10-fold cross validation directly boosts effectiveness to >91% F1 from >85%. On the other hand applying this model to the topics >41 (similar to the experiments shown in Table 2), their effectiveness measures were reduced by up to 9%. This indicates that the 10-fold cross-validated models are over-fitted to the initial topics when compared to the models generated by 3-fold cross-validation.

Table 3. C3 TFIDF-SVM results for ACAM topic mixes using 10-fold cross validation [6].

Topic	1 to 10	1 to 20	1 to 40
Arguments	202	449	813
Terms	84	157	260
F1	95%	91%	96%

To additionally ascertain the models capability to generalize we performed multiple experiments using the UKP and transferring models between corpora. Besides creating a model from the UKP arguments using 3-fold cross validation and assessing its effectiveness, we applied the UKP model on ACAM arguments and vice versa. Table 4 contains the results of these experiments.

Table 4. C3 TFIDF-SVM results with different argument stance corpora [6].

Experiment	UKP on ACAM	UKP on UKP	ACAM on UKP
Microaverage F1	48%	60%	46%
Microaverage precision	48%	60%	46%
Microaverage recall	48%	60%	46%
Pro F1	25%	64%	25%
Pro precision	16%	62%	55%
Pro recall	52%	65%	16%
Contra F1	60%	56%	58%
Contra precision	83%	57%	44%
Contra recall	47%	55%	83%

Interestingly, whenever the ACAM corpus is involved (either as source for the model or as evaluation set) the detection of contra arguments is more effective than that of pro arguments. It is noteworthy, that the UKP corpus is inherently more difficult than the ACAM. It contains shorter arguments that are only single sentences, whereas the

ACAM contained longer debate points [6]. Compared to previous state of the art results for generalizing argument stance recognition to new topics (see Sect. 2.1), C3 TFIDF-SVM's effectiveness is 7% below that of Stab et al. with the UKP corpus. Compared to SemEval2016 challenge, our results are similar even though different datasets are used. This means that our effectiveness is comparable to that of other state of the art approaches. The key difference is that C3 TFIDF-SVM can automatically generate human readable explanations for every categorization decision and has demonstrated high generalizability as it can easily be transferred to new topics. To further investigate this transferability as well as pursuing the overall C3 goal of providing TC for any application, we used the same classifier on completely different problems. The first problem is article triage, which is the task of distinguishing relevant scientific documents from irrelevant scientific documents. The second problem is offensive language detection which we performed with a German language corpus of offensive and inoffensive tweets about politics. The third problem is intent detection for digital assistants.

5.2 Using C3 TFIDF-SVM for Article Triage

Article triage is the problem of identifying relevant articles to source arguments from. To test C3 TFIDF-SVM's utility for this task, we downloaded 200 abstracts about *melanoma* as well as 200 abstracts about *leukemia* from PubMed [1]. Both topics are types of *neoplasms* (cancers), creating a TC problem with relatively similar categories. Table 5 shows the results of the best of three models created by C3 TFIDF-SVM using 3-fold cross validation.

Table 5. TFIDF-SVM PubMed article triage results for the best model using 3-fold cross validation.

Category	Precision	Recall	F1
All microaverage	81%	81%	81%
All macroaverage	82%	78%	79%
Melanoma	85%	64%	73%
Leukemia	79%	92%	85%

As shown, the best of three models created 81% microaverage F1. The worst came up to 66% microaverage F1. Similar to Argument stance recognition, one class obtained better results than the other as leukemia related abstracts are spotted with 85% F1 while melanoma related articles are identified with 73% F1.

As this is a two-class problem, the effectiveness can again be increased by logically combining results based on known category effectiveness. The effectiveness results are comparable to those obtained in the argument stance recognition task described in Sect. 5.1. Even though the C3 API caters for hyper-parameter tuning of the training process, only our C3 TFIDF-SVM default settings have been used to create these results for a completely different TC problem. Besides the automatically generated explanations, this underlines the robustness of our approach.

5.3 Using C3 TFIDF-SVM for German Language Offensive Language Detection

To further investigate the flexibility and robustness of C3 TFIDF-SVM we switched natural languages and applied it to German language offensive language detection. The GermEval 2018 Shared Task on the Identification of Offensive Language aims at the identification of offensive language in German Twitter tweets [29]. It contains a training set consisting out of 5,009 tweets. 1,778 of these contain offensive language. It additionally contains a test set of 3,532 tweets, 1,202 of which are offensive. We created a TFIDF-SVM model using the training set and evaluated in on the test set using the C3 committee service. Table 6 contains the results of this experiment.

Table 6. C3 TFIDF-SVM GermEval 2018 results using its default configuration.

Category	Precision	Recall	F1
All microaverage	60%	60%	60%
All macroaverage	56%	56%	56%
Other	70%	70%	70%
Offensive	41%	40%	41%

The results are similar to those obtained for the argument stance recognition task. In this case, the identification of non-offensive language was more effective than that of offensive language. The original GermEval 2018 competition had 51 entries. C3 TFIDF-SVM has not participated. If it had, it would have been on place 42. Notably, the creation of the model during n-fold cross validation has shown, that the detection of the OTHER category was possible with more than twice the F1 as the detection of offensive language. As C3 TFIDF-SVM is not aware of this mutual exclusiveness, a simple logical rule to only take OTHER results into account would improve C3 TFIDF-SVM's effectiveness to 70%. The winner of GermEval 2018 used a complicated committee of multiple classifiers along with manually crafted lexicons of offensive language to obtain 77% F1.

This shows, that C3 TFIDF-SVM can be used in different natural languages with minimal manual effort. The generated results are close to that of competition winners while automated human readable explanations are generated. These explanations can subsequently be taken into account when creating lexica for offensive language identification.

5.4 Using C3 TFIDF-SVM for Digital Assistant Intent Detection

A primary task for digital assistants or chat bots is intent recognition: The detection of the task that should be performed by the system [30]. This is a TC problem with more than two categories. To evaluate C3 TFIDF-SVM on this problem, we used the benchmark for digital assistants created by Coucke et al. [30]. The actual benchmark focuses on named-entity recognition within text. Therefore the benchmark contains the original text as well as one that is annotated with relevant named entities. This allows us two experiments: The first is to test TFIDF-SVM with the normal texts. The second is to replace certain text parts with the detected entity in preprocessing and evaluating C3

TFIDF-SVM with these augmented texts. Coucke et al.'s benchmark has seven different categories and is already split in a test and evaluation set. There are 400 training samples and 100 evaluation samples per category. Table 7 shows the results of a model tested by the C3 committee service.

Table 7. C3 TFIDF-SVM evaluation set results for intent detection without decoding named entities.

Category	Precision	Recall	F1
All microaverage	73%	57%	64%
All macroaverage	72%	57%	64%
Add to playlist	73%	27%	39%
Book restaurant	79%	69%	74%
Get weather	78%	54%	64%
Play music	60%	48%	53%
Rate book	82%	89%	85%
Search creative work	52%	32%	40%
Search screening event	80%	78%	79%

Even though the individual results differ from those obtained during the 3-fold cross validation process of the original model, the microaverage F1 is in both cases 64%. This highlights the robustness of C3 TFIDF-SVM's models when transferred to previously unseen data. For the next experiment, named entities were decoded. This means that for instance strings like *música libre* were replaced by their common entity *playlist*. The n-fold cross validation process using TFIDF-SVM's default settings created almost perfect results on the training set with these augmentations with 98% microaverage F1. These values were reduced down to 89% microaverage F1 when the model was evaluated with the evaluation set as shown in Table 8:

Table 8. C3 TFIDF-SVM evaluation set results for intent detection with decoded named entities.

Category	Precision	Recall	F1
All microaverage	81%	98%	89%
All macroaverage	81%	99%	89%
Add to playlist	91%	100%	95%
Book restaurant	82%	94%	88%
Get weather	81%	100%	89%
Play music	76%	99%	86%
Rate book	83%	100%	90%
Search creative work	76%	99%	86%
Search screening event	82%	98%	89%

These results show that C3 TFIDF-SVM can be combined with a preprocessing step to identify relevant named entities which strongly increase effectiveness. They also show that the system is able to generalize to different problems with more categories.

6 Understandability Survey

Different from most state of the art classifiers, C3 TFIDF-SVM generates human readable explanations for its generated results and the generated models can easily be manually inspected. The pattern to generate explanations is slightly updated from that described in the beginning of Sect. 3 to cater for more TC problems than argument stance recognition. In order to measure how well average people understand and explanation we performed a survey for which we got 26 replies. All survey participants were native German speakers. Because the generated explanations were in English, some potential participants refused to participate because the level of English in the survey was too challenging for them. About 75% of actual participants have a university degree. About 80% of the participants work in the field of information technology and computer science albeit some work in supporting functions like sales, accounting and marketing. As we deemed it the most relatable problem, we chose the explanation for the digital assistant intent detection experiment described in Sect. 5.4. C3 TFIDF-SVM was confronted with the message *"I'd like to have this track onto my Classical Relaxations playlist."* It correctly assigned it to the category *add to playlist* with and without decoded named entities.

First, the participants were asked to state if they understand the following explanation: *"This document is considered to belong to category "Add to Playlist", because it contains occurrences of the terms* classical, have, d. *In 2100 previously analyzed documents, the occurrence of these terms in their relative amounts indicated a 99.97% probability for a document to belong to category "Add to Playlist". The likelihood has to be at least 10% to be assigned to this category."*

19 out of 26 participants understood the explanation. TFIDF-SVM removes special characters during tokenization. Therefore *I'd* becomes two individual words *I* and *d*. This created a lot of confusion under the participants, many of whom stated that there is no *d* in the sentence so they suspected some kind of error.

Second, the participants were confronted with the automatically generated explanation for the case of decoded named entities. The explanation was as follows: *"This document is considered to belong to category "Add to Playlist", because it contains occurrences of the terms playlist, add, item, music. In 2100 previously analyzed documents, the occurrence of these terms in their relative amounts indicated a 99.99% probability for a document to belong to category "Add to Playlist". The likelihood has to be at least 10% to be assigned to this category."*

23 out of 26 participants understood this explanation which is a significant increase to that without decoding named entities. Interestingly, only one participant noticed, that the words *add*, *item* and *music* do not literally occur within the sentence while many pointed out, that there is no single *d* in the first explanation. This survey gave us two insights: Firstly, the automatically generated explanations are understandable by the majority of survey participants that speak the language the explanation is in. Secondly, more abstract concepts such as *item* and *music* were better understandable by survey participants.

7 Conclusions

7.1 About Topic Specific and General Terms as Features

Between our argument stance recognition experiments with the ACAM corpus we manually inspected the model's controlled vocabulary for every learned topic. This was feasible for the first 40 topics as the model contained 260 terms. C3 TFIDF-SVM identified over 1000 terms for UKP which made manual inspection unfeasible. Besides many topic specific terms, the model also contained general terms such as *incredibly*, *uncomfortable*, *radical*, *death*, *stress*, *knowledge*, *greedy*, *tragedy*, *concern*, *racist*, *cruel*, *presents*, *reason*, *justice*, *pleasure*, and *punishment*. These general terms oftentimes carry a specific sentiment. Especially the negative sentiment is more easily identifiable. In our opinion, this is the reason why the recognition of contra arguments outperformed that of pro arguments when ACAM is involved (see Sect. 5.1).

In combination with our experimental results, these terms support our hypothesis of general and topic specific argument stance features. During evaluations of the classifier with known topics, higher effectiveness results were observed than during to previously unknown topics (>85% F1 for ACAM and >60% F1 for UKP compared to >57% F1 for ACAM and >46% F1 for UKP).

Argument specificity and therefore the intersections of $F(n)$ and $F(m)$ for topics n and m can be seen as flexible because certain terms can have completely different meanings in other topics or the topics have overlapping concerns. The machine learned hyperplanes of C3 TFIDF-SVM models should compensate this by being under or above most values for this dimension. An example for such a term is *drug* which can have an entirely different meaning in the context of medical treatment and policies on drug abuse. To further test this thesis, we created new models on other topic blocks of ACAM (41–80, 81–120, and 121–160). Evaluating these models with the remaining topics of ACAM created similarly effective results than those shown in Table 2. Manual inspection of these generated models shows that they contain many more general terms like *local*, *prisoners*, *test*, *information*, *money*, *team*, *death*, *love*, *workers*, *women*, *rights*, *knowledge*, *child*, *gifts*, *unhealthy*, *exists*, *over*, *uncomfortable*, *power*, *students*, and *war*. It also contains some words indicating who is referring to what like *my*, *I*, *he*, *your*, or *she*. Some of the terms seem to be specific to more than one topic. Like the afore mentioned *drugs* or the term *marriage* which can be seen as literal or proverbial.

To provide further examples for such topic specificity, we manually assessed 400 explanations that were created by C3 TFIDF-SVM when using a model trained on the first 40 ACAM topics and tasking it with the categorization of explanations from ACAM topics 41–80. One such explanation is: *"This document is considered to belong to category "Pro", because it contains occurrences of the terms* mechanism, enough, your, cruel. *In 814 previously analyzed documents, the occurrence of these terms in their relative amounts indicated a 99.79% probability for a document to belong to category "Pro". The likelihood has to be at least 50% to be assigned to this category."*

We listed the identified features in Table 9 which differentiates them by which type of explanations they occurred in. The explanations can either be for correct or incorrect categorizations for the categories *Pro* and *Contra.*

Table 9. Specific features from ACAM explanations. The listed terms occurred in pro and contra explanations for categorizations that are either correct or incorrect.

Categorization types in which explanations terms occurred	Terms
All categorization types	Always, day, emissions, enough, governments, he, I, money, my, reason, scheme, united, your
Correct Pro categorizations	Christmas, collection, concern, condemnation, cruel, document, earth, football, heart, marriage, numerous, punishment, quote, regular, scots, St, suffer, test, treatment, truth, Wikileaks, women
All Pro categorizations but not in Contra categorizations	Age, Britain, child, economy, knowledge, report, school, single
Incorrect Pro categorizations	Added, bishop, immigrants, immigration, paying, renewable, represents, richest, round, speed, suffer, tab, workers
Correct Contra categorizations	Anne, applications, improvement, injury, looked, points, positive, relies, signature, soldiers, station
All Contra categorizations but not in Pro categorizations	Anger, exactly, fans, Korea, love, racist
Incorrect Contra categorizations	Deaths, justice, pleasure, stopped, stress, unemployment
Occurring in different categorization types	British, children, claim, compared, crisis, death, EU, Europe, experience, food, games, greedy, information, mechanisms, ideas, information, nations, north, music, mp, Obama, over, powers, president, prisoners, rights, soldiers, she, want, water

Because the text categorization uses SVMs, the combination of relative itfidf values for these terms determines whether an argument is categorized as pro or contra. This analysis still yielded some interesting insights for useful features. For instance, features that only occurred in the explanations for incorrect categorizations are likely to be topic specific for ACAM topics 1–40 but not 41–80. Two examples for such feature terms are the terms *added* and *immigrants* that only occurred in the explanations of incorrect categorizations of ACAM topics 41–80 but occurred in the explanations of correct categorizations for ACAM topics 1–40.

7.2 About Flexible, Explainable and Robust Text Categorization

Our contribution is four-fold: We firstly developed a robust text categorization classifier that can be used for argument stance recognition and achieves reasonable results for

previously unseen topics. This way, it can be used directly to identify the argument stances during the construction of the RecomRatio argument ontology. It also serves as basis for creating an ontology of argument stance features based on the generated explanations per argument.

We secondly proposed our thesis of general and topic specific features which is supported by our experiments. These can be used to structure known argument stance features as shown in Fig. 3. Such ontologies can form a basis to implement explanation support for other non-semantic applications. If individual concepts that were found as argument stance features are represented by more general hyper concepts, these could additionally be used as features for specific stances.

Our third contribution is the short development time needed to provide an application with machine learning based text categorization. The development work necessary to perform the experiments described in Sects. 5.2, 5.3 and 5.4 only consisted of data engineering to import the labeled data to C3's interface. It was below 2 h for all implementations. Neural-symbolic integrated applications are easy to develop if one only has to focus on the symbolic part as the machine learning based aspect is encapsulated behind an easy to use API that does not require any hyper-parameter tuning to produce useful results. C3 TFIDF-SVM is not only easily transferrable to new topics in the argument stance recognition problem but can also be applied to any TC problem where reasonable results can be achieved without hyper-parameter tuning.

Our fourth contribution is our work in automatically explainable explanations. The proposed approach requires no additional manually created resources besides appropriately labelled texts to create human readable explanations why certain categorizations were performed. These explanations are understood by the majority of survey participants. This makes C3 TFIDF-SVM safe to use for any enterprise grade application which has to cope with the regulatory requirement of providing explanations for the results of their applications.

Acknowledgements. This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) within the project Empfehlungsrationalisierung, Grant Number 376059226, as part of the Priority Program "Robust Argumentation Machines (RATIO)" (SPP-1999).

References

1. US National Library of Medicine National Institutes of Health pubmed.gov. https://www.ncbi.nlm.nih.gov/pubmed/. Accessed 17 Sep 2019
2. European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance); OJ L, 4 May, 2016, vol. 119, pp. 1–88 (2016)
3. Clos, J., Wiratunga, N., Massie, S.: Towards explainable text classification by jointly learning lexicon and modifier terms. In: IJCAI-17 Workshop on Explainable AI (XAI) (2017)
4. Lippi, M., Torroni, P.: Argument mining: a machine learning perspective. In: Black, E., Modgil, S., Oren, N. (eds.) TAFA 2015. LNCS (LNAI), vol. 9524, pp. 163–176. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28460-6_10

5. Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., Cherry, C: Semeval-2016 task 6: detecting stance in tweets. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 31–41 (2016)
6. Eljasik-Swoboda, T., Engel, F., Hemmje, M.: Using topic specific features for argument stance recognition. In: Proceedings of the 8th International Conference on Data Science, Technology and Applications (DATA 2019), pp. 13–22 (2019). ISBN:978-989-758-377-3
7. Mohammad, S.M., Sobhani, P., Kiritchenko, S.: Stance and sentiment in tweets. ACM Trans. Internet Technol. Argument. Soc. Media **17**, 1–23 (2016)
8. Stab, C., Miller, T., Schiller, B., Rai, P., Gurevych, I.: Cross-topic argument mining from heterogeneous sources. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2018) (2018)
9. Same Side Stance Classification. https://sameside.webis.de/. Accessed 24 Sep 2019
10. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**, 1–47 (2002)
11. Bader, S., Hitzler, P.: Dimensions of neural-symbolic integration – a structured survey. arXiv preprint arXiv:cs/0511042 (2005)
12. Swoboda, T., Kaufmann, M., Hemmje, M.: Toward cloud-based classification and annotation support. In: Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), vol. 2, pp. 131–237 (2016)
13. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943)
14. Helbig, H., Scherer, A.: Kurs 1830: Neuronale Netze. University of Hagen, Germany (2011)
15. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning – a new frontier in artificial intelligence research. In: IEEE Computational Intelligence Magazine, USA, November issue, pp. 13–18 (2010)
16. Vapnik, V.N., Chervonenkis, A.Y.: On a class of algorithms of learning pattern recognition. Framework of the Generalised Portrait Method, Об одном классе алгоритмов обучения распознаванию образов, Автоматика и телемеханика (1964)
17. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 1–27 (2011)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representation in vector space. In: Proceedings of Workshop at ICLR (2013)
19. Pennington, J., Socher, R., Manning, C.: GloBe: global vectors for word representation. In: Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)
20. Zanzotto, F.M., Korkontzelos, I., Fallucchi, F., Manandhar, S.: Estimating linear models for compositional distributed semantics. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 1263–1271 (2010)
21. Kusner, M.J., Sun, Y., Kolkin, N., Weinberger, K.Q.: From word embeddings to document distances. In: Proceedings of the 32nd International Conference on Machine Learning (2015)
22. Dai, X., Bikdash, M., Meyer, M.: From social media to public health surveillance: word embedding based clustering method for twitter classification. In: Proceedings of SoutheastCon, pp. 1–7 (2017). https://doi.org/10.1109/secon.2017.7925400
23. Eljasik-Swoboda, T., Kaufmann, M., Hemmje, M.: No target function classifier – fast unsupervised text categorization using semantic spaces. In: Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018), pp. 35–46 (2018)
24. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805v2 (2019)
25. Wolff, E.: Microservices – Flexible Software Architecture. Pearson Education, USA (2017)
26. Dropwizard: Production-ready, out of the box. https://dropwizard.io. Accessed 12 Sep 2019
27. Enterprise Container Platform | Docker. https://www.docker.com/. Accessed 30 Sep 2019

28. Peldszus, A.: An annotated corpus of argumentative microtexts. https://github.com/peldszus/arg-microtexts. Accessed 15 Mar 2019
29. Wiegand, M., Siegel, M., Ruppenhofer, J.: Overview of the GermEval 2018 shared task on the identification of offensive language. In: Proceedings of the GermEval, Vienna, Austria (2018)
30. Coucke, A., et al.: Snipts voice platform, an embedded spoken language understanding system for private-by-design voice interfaces. arXiv:1805.10190 (2018)