Using Topic Specific Features for Argument Stance Recognition

Tobias Eljasik-Swoboda[1][a], Felix Engel[2][b] and Matthias Hemmje[2][c]
[1]*Faculty of Mathematics and Computer Science, University of Hagen, Hagen, Germany*
[2]*FTK e.v. Forschungsinstitut für Telekommunikation und Kooperation, Dortmund, Germany*

Keywords: Argument Stance Detection, Explainability, Machine Learning, Trainer-athlete Pattern, Ontology Creation, Support Vector Machines, Text Analytics, Architectural Concepts.

Abstract: Argument detection and its representation through ontologies are important parts of today's attempt in automated recognition and processing of useful information in the vast amount of constantly produced data. However, due to the highly complex nature of an argument and its characteristics, its automated recognition is hard to implement. Given this overall challenge, as part of the objectives of the RecomRatio project, we are interested in the traceable, automated stance detection of arguments, to enable the construction of explainable pro/con argument ontologies. In our research, we design and evaluate an explainable machine learning based classifier, trained on two publicly available data sets. The evaluation results proved that explainable argument stance recognition is possible with up to .96 F1 when working within the same set of topics and .6 F1 when working with entirely different topics. This informed our hypothesis, that there are two sets of features in argument stance recognition: General features and topic specific features.

1 INTRODUCTION

The RecomRatio project seeks to implement an information system that supports medical professionals by recommending treatment options and supplying rational arguments why a specific treatment is suggested. The recommendation will be based on an argument ontology that compares pro and contra arguments to given topics in medicine. Basis for the ontology instantiation are specific information units, extracted from publicly available data sets, as e.g. provided by PubMed or other similar sources (US National Library of Medicine, 2018). The necessity to explain the recommendations is important because of two reasons. Firstly, medical practitioners have more trust in the system's recommendations if they are shown the reasons for the recommendation. Secondly, the General Data Protection Regulation (GDPR) contains a right to explanation, which demands explanations for the results of machine learning and artificial intelligence systems if they impact an EU citizen (EU, 2016). This requirement is relatively new and has not been an aspect of machine learning and artificial intelligence research until recently (Clos et al., 2017).

An essential task in generating explanations for recommendations is to reliably detect the stance of an argument, to correctly represent it in the ontology and to provide further information to support the traceability of reasons that led to the specific classification. Hence, our work is in the area of Argument Mining (AM). More specifically in the task of determining whether a statement is a pro argument supporting a given topic or a contra argument against this topic. In order to explain why an argument was classified in a specific way, one needs a classification system that grants a high degree of insight into its internal processes. To learn stance detection we applied an adoption of the LibSVM classifier tool to a set of arguments (see section 2.4). The main hypothesis that we intend to analyze is:

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).*

[a] https://orcid.org/0000-0003-2464-8461
[b] https://orcid.org/0000-0002-3060-7052
[c] https://orcid.org/0000-0001-8293-2802

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 contribution, supporting our hypothesis is structured as follows. Firstly we introduce the relevant state of the art in the fields of Argument Mining, Neural-Symbolic Integration, Machine Learning based Text Classification, Feature Assessment and evaluation metrics. Secondly, we introduce the experimental setup and applied methodology. Thirdly, we document and analyze the experimental results. Last but not least we discuss our outcome and provide an overview of future work.

2 STATE OF THE ART

2.1 Argument Mining (AM)

Generally speaking, research in Argument Mining aims to automate the process of detecting arguments in large quantities of text. Essentially AM brings together a quite broad set of different disciplines like Artificial Intelligence, Natural Language Processing and Computational Linguistics. Seeking for answers to challenges related to "*… natural language processing and understanding, information extraction, feature discovery and discourse analysis*" (Lippi and Torroni, 2015).

A specific challenge within AM is the detection of the stance of an argument. Essentially meaning, the detection if an argument is a pro or contra argument given a specific topic. Various stance detection approaches have been evaluated e.g. on the Semeval16 conference (Mohammad et al., 2016).

2.2 Neural-Symbolic Integration

Neural-Symbolic Integration is the fundamental idea to merge symbolic knowledge representation, for example expressed in ontologies, with neural network based supervised learning (Bader and Hitzler, 2016).

Reasons to adopt this approach are to utilize benefits of both approaches: Symbolic, or semantic systems are logic-based, declarative and explicitly model how humans think. Neural networks on the other hand are much more tolerant against noise and more robust when working with previously unseen data. Neural networks are trained on example data and automatically generate their functionality during training. This way they express the regularities of their training set but not explicit human-generated knowledge. Neural-Symbolic Integration attempts to benefit from the strengths of both approaches.

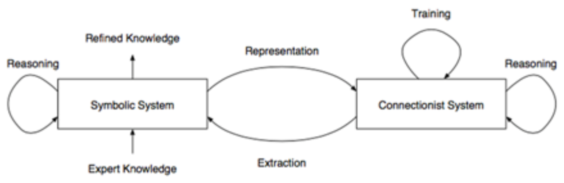

Figure 1: Neural-Symbolic Integration (Bader and Hitzler, 2016).

Even though the name implies neural networks, Neural-Symbolic Integration is not necessarily limited to neural networks but can be used with any supervised machine learning approach one can easily integrate with a symbolic, semantic application.

2.3 Cloud Classifier Committee

The Cloud Classifier Committee (C3) is a suite of microservices that implement different machine learning approaches making it feasible to combine their results in a committee fashion. C3 implements a modern and flexible microservice-oriented architecture (Swoboda et al., 2016); (Wolf, 2017). The goal of these microservices is to make Text Categorization (TC) easily available for any type of application, including explainable AM pro / contra recognition. C3 exposes a number of endpoints on a REST/JSON interface which any network capable application can connect to. This makes integrating TC techniques into arbitrary applications as difficult as accessing an external database. C3 uniquely enables neural-symbolic integration because it uses a common API for any supervised learning TC algorithm that a symbolic application can interface with. We use Support Vector Machine (see next sub-section) based microservices because their model is easy to explain in comparison to more obscure models such as neural networks. In TC, documents can be of arbitrary length, ranging from a few words, over sentences to lengthy texts. For us, a document is a string of different words of arbitrary length.

To the best of our knowledge, there are no best practices and design patterns to enable machine learning based text categorization in microservice-oriented architectures. We therefore developed the trainer-athlete pattern, which is adopted by C3. Within the *trainer-athlete* pattern there are two roles of services: The trainer learns a model by applying supervised learning algorithms to the available corpus of documents, categories, and their assignments (the target function). After the model is learned, it is made available as JSON file using a specific endpoint of the Trainer. The Athlete service needs such a model to function and must be configured with it after starting the service.

This pattern is loosely based on the Command Query Responsibility Segregation (CQRS) pattern, which dictates that different microservices are responsible for reading or writing data (Wolff, 2017). Trainer- as well as athlete microservices can write data. The difference is, that the fundamental model used by athletes must first be generated and written by a trainer.

While using a model in a specific application can be time critical, the computation of the model by the trainer service can be much lengthier. As soon as the model is computed, athlete services can be scaled out as required by the application. Launching additional containers running the athlete service and initializing them with the same model can easily perform this feat.

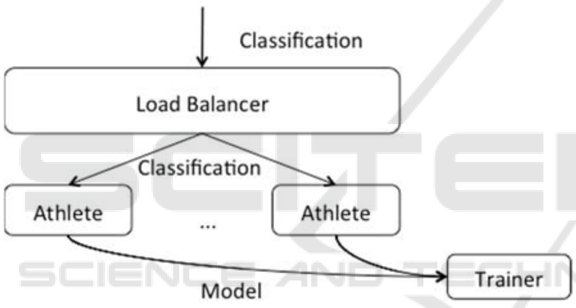

Figure 2: The trainer/athlete pattern for microservice-oriented machine learning.

Microservices, especially when flexibly packed e.g. using Docker, have advantages when compared to legacy architectures in regards to the GDPR. Implementing a machine learning based application is a non-trivial task. Outsourcing this task to external providers can be difficult for technical as well as jurisdictional reasons. Simply adding microservices to an environment already hosting sensitive data eliminates such difficulties as well as lengthy data transfer operations.

To do so, the C3 API uses common endpoints for documents, categories, the relationships between categories, the target function and categorization results. The configuration of hyper-parameters as well as generated (trainer) or utilized (athlete) models are algorithm specific. Which algorithm / microservice can use which model is controlled by embedded metadata objects that are part of every configuration and model JSON. Additionally, every C3 service has a metadata endpoint which returns all metadata about the service itself easing orchestration.

In addition to this API, every C3 service includes a web-based graphical user interface. There is also a scriptable command line interface utility further enhancing the systems flexibility.

2.4 Machine Learning based Text Categorization

Text Categorization (TC) is the task of automatically assigning documents to predetermined categories. The target function defines which document belongs to which category. A classifier is any piece of software, which approximates the target function as effectively as possible (Sebastiani, 2002). There are two fundamental approaches to tackle TC: Rule-based or machine learning based. The first approach requires expert knowledge and manual labor to explicitly capture said experts experience. The second one requires a target function to learn from and has been reported to be more robust to noise and previously unseen data. The C3 API supports microservices implementing both approaches.

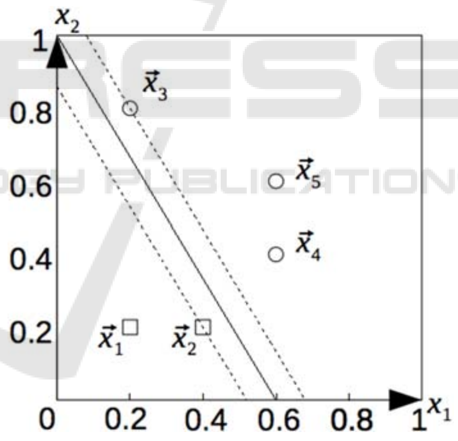

Figure 3: Support Vector Machine example.

Our work focuses on a Support Vector Machine (SVM) based approach combined with an automatically created controlled vocabulary generating bag of words (BOW) based document representations.

We chose SVMs because they are relatively easy to explain. Like most machine learning based classifiers, SVMs expect their document expressed as a vector modeling the document. After having these points representing the documents, SVMs use optimization algorithms to find a hyperplane bisecting the space occupied by the document representing points so that the documents belonging to one category are on one side of the hyperplanes, while the others are on the other. It does so by maximizing the margin between the closest vector and the hyperplane (see figure 3).

LibSVM can work with different kernels. Kernels essentially substitute the formula to compute the dot product used in the computation of distances. This means, that not only a linear hyperplane, but more sophisticated functions, like the radial basis function (RBF) can be used to tell categories from each other.

LibSVM is a standard library implementing SVMs (Chang and Lin, 2011). We used it as core for a C3 trainer and athlete. Generating feature vectors from documents is referred to as feature extraction. Selecting the most representative features for a specific document is called feature selection.

Our classifier performs feature extraction and selection, using the bag of words (BOW) model. In the BOW model, a controlled vocabulary tells relevant words from irrelevant word. How the controlled vocabulary is generated and the actually feature extraction and selection is performed is subject of section 3.3.

This doesn’t mean, that the same BOW model cannot be used for feature extraction and selection for a neural network based or completely different learner algorithm.

Besides this SVM approach, Clos et al. have proposed the explainable, lexicon based ReLexNet classifier (Clos et al., 2017). Lexicon based classifiers are comparatively simple constructs, that are trained by generating an association matrix between individual terms and classes. The more terms of high association exist, the more likely a document belongs to a category. Besides these association terms themselves, there are strength modifiers and negator terms that are looked for in the words accompanying the association terms. Negators multiply the term’s association with -1 while strength modifiers like *very* can boost a term’s association value by e.g. 1.3. Besides computing the class, these association terms can be used to explain, why a specific class was chosen. There are three basic approaches to generating the required lexica: Firstly, they can be manually created. Secondly, one can base it on ontologies by manually specifying class-indicating terms and using ontologies to find terms equally as specific. Thirdly, one can use the conditional probability of a term occurring in a specific document within the corpus to extrapolate to the probability of the term specifying the category.

Even though being an intriguing, explainable TC approach, we have not yet implemented lexicon based classifiers in C3 because it requires manually described modifiers. The primary goal of C3 is to minimize manually provided requirements to use TC.

2.5 Feature Assessment and Evaluation Metrics

Feature assessment, extraction, selection and rating of classification approaches in this publication lean on concepts from the Information Retrieval community. The so-called *Term Frequency Inverse Document Frequency* (TFIDF) is a statistic value that reflects the importance of a term in a document, taking into account its occurrence in the whole data set. Essentially TFIDF rates the number of times a specific term is contained in a document (normalized by the frequency the term occurs in the text), in proportion to the logarithm of the proportion between the number of documents in the database to the number of documents containing the term. TFIDF is used e.g. as a weighting schema to detect words with a high discriminative value (Salton and McGill, 1983).

To evaluate the ability of a retrieval system in a specific retrieval task, the basic evaluation metrics are *precision* and *recall*. Precision is the proportion between the intersection of relevant and retrieved documents to retrieved documents. Recall, in turn is calculated as the proportion of the intersection between relevant documents and retrieved documents to relevant documents. On base of these core measures several further metrics have been proposed. E.g. the F1 score that measures the accuracy of a retrieval system, by calculating the harmonic mean between precision and recall.

Introduced for information retrieval, these evaluation metrics have been widely applied in the evaluation of other fields like text categorization, natural language processing or argument detection. There are different averaging methods for these effectiveness measures (Sebastiani, 2002): In macroaveraging, the results of individual categories are averaged. In microaveraging, the individual true positive, false positive, and false negative results of all categories are summed up and then used to calculate the effectiveness measures.

3 METHODOLOGY AND EXPERIMENTAL SETUP

Having G and $F(t)$ as controlled vocabulary, one can automatically explain classification decisions with 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"*.

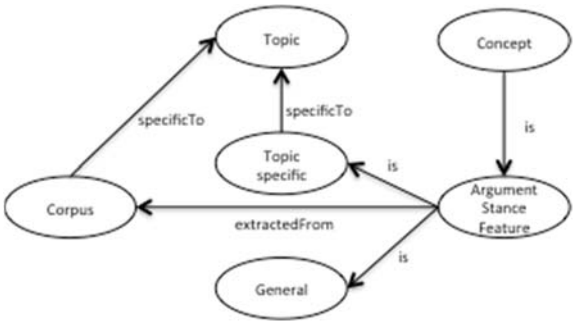

Figure 4: Structure of a topic specific argument stance feature ontology.

The pro/contra argumentation ontology could therefore decide how this argument relates to a contra argument by referencing the terms of G and $F(t)$ as *"argument stance features"*. These terms are potential existing concepts in topic specific ontologies.

Further reasoning can be performed by the relationships between the concepts that serve as argument stance features. This is similar to a neural-symbolic integrated application.

Therefore, our goal is to generate such a controlled vocabulary as part of a machine-learned model for argument stance recognition. A second goal is to determine if the terms within the controlled vocabulary are part of $F(t)$ or G.

3.1 Argument Corpus

We evaluated our system using an *annotated corpus of argumentive microtexts* (ACAM) (Peldszus, 2016). In total, the corpus contains 25,351 argument pairs from arguments about 795 different topics. However, the corpus is a composition of two corpora. On the one hand, a corpus that is composed of a set of arguments collected during an experiment that involves 23 subjects. Participants are discussing controversial topics in German. The resulting arguments have then been professionally translated to English. On the other hand, a corpus that contains arguments written by Andreas Peldszus mainly used for teaching activities.

In order to additionally validate our classifier, we used the UKP Sentential Argument Mining Corpus (Stab et al., 2018). Stab et al. sourced this corpus by querying Google to eight controversially discussed topics. They then automatically collected sentences from Google previews and had crowd workers annotate them using three classes: No Argument, Argument For, and Argument Against. For our purpose, we filtered the No Argument values in order to obtain pro arguments (Argument For) and contra arguments (Argument Against). It is noteworthy, that C3 can also be used to tell arguments from non-arguments and also annotate these decisions with explanations

3.2 Service Implementation

Two C3 classifier microservice implementations have been used for our experiments: The first one is a classifier committee service. Its actual purpose is to combine the results of multiple classifiers. We use it to externally and automatically evaluate the results of other C3 classifiers, as it weights the effectiveness of the external classifiers to determine its own model. The other C3 classifier is built on LibSVM and uses TFIDF for feature extraction (see previous sections). We chose Support Vector Machines over Neural Network based algorithms because they are computationally lightweight and can be intuitively explained: A document was categorized in a certain way, because its features are above the hyperplane defining the category. This is especially easy to understand, when linear hyperplanes are used. We chose this combination of techniques over Naïve Bayes or Decision Rule classification, because literature suggests results of higher effectiveness and no manual labor to define decision rules are required (Sebastiani, 2002)(Swoboda et al., 2016). This makes our approach not strictly neural-symbolic nevertheless providing the same benefits.

3.3 Training

In detail, the C3 TFIDF SVM classifier trainer works as follows: In a first step, the TFIDF value for every term (t_k) existing in any available document (TS) is computed. This requires counting every word occurrence within every document. A principal goal of C3 is to eliminate the need for additional resources, such a stop word lists and key phrase concept definitions. Therefore no such filtering of terms within the documents takes place. A benefit of the TFIDF equation is, that if a term t_k occurs in all or almost all documents of the training set, its log goes towards zero. Therefore, stop words that occur in most documents automatically have low TFIDF values.

$$tfidf(t_k,d_i)=\#(t_k,d_i)*\log(|TS|/\#TS(t_k)) \quad (1)$$

After computing all TFIDF values, the trainer determines which terms are used for feature extraction within the athlete services. To do so, the trainer identifies a configurable number of terms with the highest TFIDF values across all documents belonging to a given category. We used the default value of 20, which means that it identifies the 40 TFIDF values most representative for pro- and contra arguments. In the next step, this list is augmented with the single highest TFIDF term of every document. These terms are only added to this automatically created controlled vocabulary, if they are not already in it. This scheme eliminates the necessity to manually define a controlled vocabulary, is relevant for any natural language and greatly accelerates the adoption of this machine learning based TC technique for argument stance detection. Additionally, the identified feature terms serve as basis for the construction of the argument ontology within the RecomRatio project.

For training and subsequent productive classification, feature vectors are passed to a LibSVM based SVM. LibSVM requires its feature vectors to be normalized per dimension. This means that for example in all feature vectors the scalars representing the term *good* need to have a combined length of 1. For every document, a feature vector is generated containing entries for all terms that are part of the automatically generated controlled vocabulary. LibSVM expects sparse vectors in the form of arrays, which contain tupels consisting of the dimension and value for every feature. The dimension is specified by the term's id within the controlled vocabulary. Its value is its normalized TFIDF value. In order to compute the normalized TFIDF value for previously unseen documents, the document frequency of the term within the training set ($\#TS(t_k)$) is stored alongside the term in the model as well as the size of the overall training set ($|TS|$). Additionally the model contains the sum of the squares of all TFIDF values for every term of the controlled vocabulary in the training set. This is necessary to perform dimensional normalization of the values when extracting features from a new document.

Besides the ability to express a document as a feature vector suited for LibSVM, the JSON based model is human readable so that one can read which terms are taken into consideration when performing classification. This is a necessity to render the system explainable and determine the sets G and $F(t)$.

Another important step for working with LibSVM is to perform weighting of the individual categories. LibSVM weights the error for each category differently in the training process. The results are best, if the weights express the ratio of available training samples for every category. Like the controlled vocabulary, the weighing is automatically determined by the C3 TFIDF SVM microservice, so that no additional resources and configurations are required.

3.4 Evaluation

After feature extraction, the trainer applies n-fold cross-validation to find the best deciding hyperplanes using a linear kernel. In n-fold cross-validation the available documents (in this case arguments) are split up into *n* subsets. In *n* different training sessions, individual classifiers are generated using different *n*-1 subsets of documents to learn from. After training, the classifier is evaluated with the document subset not used during training. Its performance is measured in the before mentioned effectiveness values.

In previous experiments, we found that linear kernels work better than RBF kernels when using our feature extraction approach. During this training, we used LibSVM's default settings. As they are fine-tunable, they can be parameterized in C3. Without additional configuration, C3 uses the libraries defaults, which we did in our experiments. It is noteworthy, that the optimization of classifier hyper-parameters is a problem in itself that can take excessive experimentation in order to maximize effectiveness. Aubakirov et al. implemented a distributed genetic algorithm for that specific task (Aubakirov, 2018). Albeit generating highly effective classifiers for the dataset they are trained on, this approach can lead to overfitting to the training data. We therefore chose to commonly use the default parameters.

The SVM hyperplanes that performed best during cross validation are stored in the trainers' model. The trainer automatically logs the performance in terms of precision, recall and F1 in microaverage, macroaverage as well as for every category. As default configuration, C3 selects the model with the highest microaverage F1. This can also be parameterized. As previously mentioned, we used a classifier committee trainer service to evaluate the effectiveness of one model on another set of arguments by initializing an athlete service with said model and performing the training of the committee trainer which automatically called the TFIDF-SVM athlete's C3-API. It automatically logs the performance of the utilized athletes in order to ascertain their effectiveness for optimizing its weighting function.

4 RESULTS: GENERAL AND TOPIC SPECIFIC FEATURES FOR ARGUMENT STANCE RECOGNITION

In our first experiment, we trained a model on 5 individual topics of the ACAM. These topics were wildly mixed ranging from a discussion about record stores vs. internet-bought, downloaded music collections to the statement that the United States policy on illegal immigration should focus on attrition through enforcement rather than amnesty. Doing so, the trainer achieved F1 values of up to .96. The F1 for contra arguments was oftentimes at 1 meaning perfect results. As expected, these models contained primarily topic specific terms like *cd*, *record*, and *collection* for the musical discussion or *violence*, *national*, or *supporters* for immigration policy. The next step was to train the classifier on larger sets of topics using C3's default 3-fold cross validation.

After training the microservice with arguments from up to 40 topics, a drawback of the automatically generated TFIDF feature extraction approach became apparent: Its time requirements grow almost quadratically. The amount of features grows only slightly sub-linearly to the amount of arguments, because the most representative term for a document (in TFIDF) oftentimes is not already part of the automatically generated controlled vocabulary. This means that per new document (or argument), a new term is added to the BOW model. This increases the length of document representing vectors. As obviously there is a new vector per document (or argument), the size of the training set for the SVM grows almost quadratically, drastically increasing training time for larger datasets. As training a classifier with all 795 topics was unfeasible on our available hardware, we computed models for the first 10, 20 and 40 topics. Their size and evaluation results are shown in table 1. The F1 values are promising even though the almost quadratic growth in size is apparent.

To see if this model generalizes well, we used the already computed model generated from the first 40 topics to initialize an athlete service. We then used a committee trainer service to evaluate the effectiveness of the athlete by querying it with the remaining topics. Besides enabling us to evaluate our approach with the remaining topics, this also shows that we can use a model generated from less than 5% of all topics and generalize to all remaining topics of the ACAM. As expected, we obtained less effective results than in the previous experiments (see table 2). Like in the previous experiments, the effectiveness to recognize contra arguments was much better that that to detect pro arguments.

It is noteworthy, that the SVM model doesn't have the information that both categories (pro and contra) are mutually exclusive. As we know the models effectiveness for identifying pro and contra arguments, we can use this information in order to boost the systems overall effectiveness. For instance, 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). This will also increase the system's overall F1 values.

Table 1: 3-Fold cross validation results when training from the first topics.

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

Table 2: Effectiveness of a model trained on topics 1-40.

Topics:	41 to 120	41 to 200	41 to 795 (all topics)
F1	.6	.57	.57
Precision	.6	.57	.57
Recall	.6	.57	.57
Pro F1	.23	.21	.21
Pro Precision	.17	.16	.15
Pro Recall	.35	.34	.35
Contra F1	.73	.7	.71
Contra Precision	.83	.82	.83
Contra Recall	.64	.62	.62

Using this information in a symbolic system forms a fundamental neural-symbolic integration albeit no neural networks were used. Obviously, neural network based C3 classifiers can be used in the same fashion.

In all previous experiments, the trainer performed 3-fold cross-validation, which is C3's default configuration. An additional round of experimentation was performed using 10-fold cross-validation. Switching to 10-fold cross-validation directly boosted effectiveness for the best model during training (see table 3). On the other hand, the effectiveness when applying this model to the remaining topics was reduced in all effectiveness measures for 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.

We then performed multiple experiments, creating a model based on the UKP corpus and evaluating it with the ACAM and vice versa (see table 4). As expected, the effectiveness measures for evaluating the UKP based model on the UKP corpus (during 3-fold cross-validation) were better than evaluating the UKP model on the ACAM corpus or vice versa. 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. Stab et al. reported F1 values of .67 in their experiments using an attention based neural network for the task (Stab et al., 2018).

Even though our system is slightly less effective, it can be adapted to different tasks in a rapid fashion. It also allows for an easily understandable explanation of its results. This easy explanation aids us in the construction of the pro- and contra argument ontology within the RecomRatio project. Therefore we did not benchmark any other approaches used in literature with our datasets.

For comparison: The winner of the Semeval 16 stance detection challenge produced .68 F1 results while the average was .62 when working with previously seen topics (Mohammad et al., 2016). When detecting stance for a previously unseen topic, the winner of the same challenge had .56 F1 with the competition average at .37. Even though different datasets were used, this means that our results are comparable to those reported in literature.

Table 3: Effectiveness for 10-fold cross-validation.

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

Table 4: Experiments with different corpora.

Experiment	UKP on ACAM	UKP on UKP	ACAM on UKP
F1	.48	.6	.46
Precision	.48	.6	.46
Recall	.48	.6	.46
Pro F1	.25	.64	.25
Pro Precision	.16	.62	.55
Pro Recall	.52	.65	.16
Contra F1	.6	.56	.58
Contra Precision	.83	.57	.44
Contra Recall	.47	.55	.83

5 DISCUSSION

Between our experiments, we manually inspected the model's controlled vocabulary for every learned topic. This was feasible for the first 40 topics of ACAM containing up to 260 terms. The system identified over 1000 terms per topic in the UKP model, making manual inspection impracticable.

Besides the afore mentioned topic specific terms, the model also contained more general terms such as *incredibly*, *uncomfortable*, *radical*, *death*, *stress*, *knowledge*, *greedy*, *tragedy*, *concern*, *racist*, *cruel*, *presents*, *reason*, *justice*, *pleasure*, and *punishment*.

These more general terms often carry a specific sentiment. Especially a negative sentiment is more easily identifiable. We think that this is the reason why the recognition of contra arguments tends to outperform the recognition of pro arguments in ACAM.

These terms and our experimental results support our hypothesis of general and topic specific terms as features. When evaluating the classifier with arguments about the same topic or topic collection it was trained on, high effectiveness results were observed (>.85 F1 for ACAM, >.6 F1 for UKP). When switching to different topics, effectiveness is decreased (>.57 F1 for ACAM, >.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. In any case, the machine learned model, in our classifier the hyperplanes, should compensate for this, e.g. by being under or above most values for this dimension. An example for this is the term drug. It is entirely different in the context of medical treatment than in the context of policies on drug abuse.

In order to further support our thesis, we generated models on other topic blocks of ACAM (41-80, 81-120, and 121-160). We then evaluated these against the remaining topic blocks obtaining similar results to those shown in table 2. Like before, we manually assessed the controlled vocabularies generated by these trainers to identify terms that occur in more than one of them.

The set of terms that occur in the controlled vocabularies of different topic mixes (e.g. terms identified in topics 1-40, 41-80 and 81-120) contains 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.

6 OUTLOOK

Our contribution is three-fold: We firstly developed a classifier for argument-stance recognition, which is explainable by the features it uses. As such, it serves as basis for creating an ontology of argument stance features that aids in the construction of the overall argument ontology of the RecomRatio project. We secondly proposed our thesis of general and topic specific features, which is supported by our experiments. These can be of further use to structure the argument stance feature ontology and connect it to topic specific ontologies. Such integrated topic specific argument stance ontologies form the basis for explainability in a non-semantic application.

Secondarily, the short development time needed to generate these results proves the versatility of the C3 microservices and its utilized trainer/athlete pattern, which we see as our third contribution. 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. This is includes the design philosophy to automatically generate model parameters, e.g. the category weights and controlled vocabulary.

In future experiments, we are aiming to precisely identify general and topic specific argument stance features for the specific knowledge domains of our research project. This means, that we aim to extend existing medical ontologies with an explainability-dimension that specifies, whether concepts are features for certain classification algorithm concepts. These algorithms are in turn linked to explanation templates and the corpora from which their model was generated. To the best of our knowledge, existing medical ontologies lack this explainability angle, which we identify as important extension.

Based on these ontologies, additional lexicon based classifiers including modifier terms can be created and put into a committee with the proposed TFIDF-SVM classifier.

As the new lexicon-based system as well as our existing system are explainable, the committee decision can also be explained by referring to the individual classifications and mentioning, how well these classifiers performed in evaluations. Additionally, evaluation results for the individual mutually exclusive classes can be taken into account when computing combined classification decisions.

We also intent to generalize our approach to determine arguments from non-arguments as we expect that there are also sets of general and topic specific words for this task. Comparing them to those identified in argument stance recognition can further aid in developing a feature ontology, capable of not only explaining why something is considered pro- or contra but also why it is an argument at all.

ACKNOWLEDGEMENTS

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) within the project Empfehlungsrationalisierung, Grant Number 643018, as part of the Priority Program "Robust Argumentation Machines (RATIO)" (SPP-1999).

REFERENCES

Aubakirov, S., Trigo, P., Ahmed-Zaki, D., Distributed Optimization of Classifier Committee Hyperparameters, Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018), pages 171-179, 2018.

Bader, S., Hitzler, P. (2005). Dimensions of neural-symbolic integration-a structured survey. *arXiv preprint cs/0511042.*

Chang, C., Lin, C., LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, volume 2, issue 3, pp 27:1 –27:27, 2011.

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.*

Regulation (EU) 2016/679 of the 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 119, 4.5.2016, p. 1–88.

Lippi, M., & Torroni, P. (2015, July). Argument mining: A machine learning perspective. In *International Workshop on Theorie and Applications of Formal Argumentation* (pp. 163-176). Springer, Cham.

Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., & Cherry, C. (2016). Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 31-41).

Peldszus, A., An annotated corpus of argumentative microtexts, https://githuzb.com/peldszus/arg-microtexts, November 2016.
Salton, G., McGill, M., *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983
Sebastiani, F., Machine Learning in Automated Text Categorization, ACM Computing Surveys, Vol 34, pp. 1-47, 2002
Stab, C., Moller, T., Schiller, B., Rai, P., Gurevych, I., Cross-topic argument mining from heterogeneous sources, Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2018), October 2018.
Swoboda, T., Kaufmann, M., Hemmje, M. L., Toward Cloud-based Classification and Annotation Support, Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) – Volume 2, pp. 131-237, 2016.
US National Library of Medicine National Institutes of Health pubmed.gov; https://www.ncbi.nlm.nih.gov/pubmed/ accessed March 11, 2018.
Wolff, E., Microservices – Flexible Software Architecture, Pearson Education, ISBN-13: 978-0-134-60241-7, 2017.