Exploiting Entity Information for Stream Classification over a Stream of Reviews

Christian Beyer
Otto-von-Guericke Univ. Magdeburg
Germany
christian.beyer@ovgu.de

Vishnu Unnikrishnan
Otto-von-Guericke Univ. Magdeburg
Germany
vishnu.unnikrishnan@ovgu.de

Uli Niemann
Otto-von-Guericke Univ. Magdeburg
Germany
uli.niemann@ovgu.de

Pawel Matuszyk
Otto-von-Guericke Univ. Magdeburg
Germany
matuszyk.pawel@googlemail.com

Eirini Ntoutsi
Leibniz Univ. Hannover
Germany
ntoutsi@kbs.uni-hannover.de

Myra Spiliopoulou
Otto-von-Guericke Univ. Magdeburg
Germany
myra@ovgu.de

ABSTRACT

Opinion stream classification algorithms adapt the model to the arriving review texts and, depending on the forgetting scheme, reduce the contribution old reviews have upon the model. Reviews are assumed independent, and information on the *entity* to which a review refers, i.e. to the opinion target, is thereby ignored. This implies that the prediction of a review's label is based more on reviews referring to other, more popular or simply more recently inspected entities, while reviews referring to the same entity might be ignored as too old. In this study, we enforce that the reviews to each entity are taken into account for learning, adaption and forgetting.

We split the original stream to substreams, each substream comprised by the reviews referring to the same *entity* (opinion target). This allows us to deal with differences in the speed of each substream and to exploit the impact of the entity itself on the labels of the reviews referring to it. For this constellation of substreams we propose a pair of two voting classifiers, one being the global, "entity-ignorant" classifier trained on the whole stream of reviews, the other one consisting of one "entity-centric" classifier per entity. We show that the entity-ignorant classifier contributes most for entities with very few reviews, i.e. during the cold-start, while the entity-centric classifiers contribute most after acquiring enough information on the corresponding entities. We study our approach on a stream of product reviews, show that our ensemble improves the performance of its members, and we discuss the conditions under which one member contributes more than the other.

CCS CONCEPTS

• **Theory of computation** → *Streaming models*; Models of learning;

KEYWORDS

Entity-Centric Learning, Stream Classification, Document Prediction

ACM Reference Format:
Christian Beyer, Vishnu Unnikrishnan, Uli Niemann, Pawel Matuszyk, Eirini Ntoutsi, and Myra Spiliopoulou. 2019. Exploiting Entity Information for Stream Classification over a Stream of Reviews. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3297280.3297333

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297333

1 INTRODUCTION

Stream classification algorithms predict the label of each observation by learning a model of past labeled observations and adapting this model as new labels arrive. When applying this mode of learning and adaptation to predict the ratings of product reviews, though, one must take into account that reviews on a product are not independent observations, as expected by a stream classifier. Rather, the reviews depend on the product itself; this may influence the rating given, the choice of words used, even the likelihood of seeing a review for the product or of observing drift. In this study, we perform a first step towards predicting review ratings from the products' perspective, by investigating how an ensemble of dedicated, product-specific learners contributes to prediction quality.

A product is a specific example of an *entity* that stands in an 1-to-n relationship with observations concerning that entity. The measurements by each of many pollution sensors in a region, the purchase transactions of a customer or the opinionated postings of a social network participant are further examples. In (multivariate) time series analysis, the relationship between entity and observation is naturally taken into account, since observations from e.g. different sensors are never mixed. However, time series are not streams: the prediction of the next observation's label, given the observation's content and the past labeled observations, is a different task than predicting the next value or multivariate vector in a time series. *Awareness* of the entity-observation relationship was very recently proposed in [1] but their approach uses simple entity-centric models which solely relied on an entities past labels and they did not combine their entity-centric and entity-ignorant models to form a prediction.

Our approach for entity-aware stream learning encompasses an entity-ignorant classifier, which learns on all instances indiscriminantly, and a set of entity-centric classifiers, each of which learns on the instances appertaining to this entity only. We combine these two types of classifier in a weighting and voting scheme, and investigate under which conditions entity-centric learning brings advantage towards the naive scheme of ignoring the entities.

The paper is organized as follows. In the next section we present related work. We then describe our approach, as well as an evaluation framework designed to measure the contribution of the new type of model learning to performance improvement. Subsequently, we report on our experiments with two streams of reviews. In the last section, we summarize our findings and close with an outlook.

2 RELATED WORK

Relevant to our work are stream algorithms that waive the assumption of independence among arriving reviews and ratings of products, as well as stream algorithms that exploit the association between rating and product for learning and for forgetting. Ensemble learning on text streams is a subject of further relevance.

2.1 Entity awareness

An early example of awareness concerning the relationship between an entity, like a product, and a temporary phenomenon associated with it, appears as in definitions of 'context': In [5], Hong et al. state that 'Context is any information that can be used to characterize the situation of an entity.' In [8], Oku et al. show that context-aware features lead to better recommender performance. While context-aware learning places an entity in a given (predefined or learned) context, we focus on the relationship between an entity and each of the infinite number of observations referring to it, and use this relationship to inform the model over the observations.

Closer to our notion of entity-centering over a stream of observations are the works of [2, 7, 10] and [1]. In [2], Fafalios et al. propose a set of entity-centric measures like popularity and sentimentality extracted from the Twitter social stream to characterize individual entities over time. In [7], Melidis et al. propose a text stream classification algorithm that detects and handles drifts in the feature space: they collect historical information for each feature and then use an ensemble that predicts a feature's future value considering a variety of feature trends, namely, regular, seasonal, autoregressive and short-bursting feature trends each captured by a base-predictor. In that context, a feature can be perceived as an entity, whose history is being exploited.

In [10], all observations on an entity are linked into an infinite timeseries: Unnikrishnan et al. exploit the similarity between entities to inform a predictor that forecasts the labels of observations in the near and the far future. Although they model a stream of observations, as we do, they assume that there is only an initial set of labels available, so their goal is to predict future labels for as far as possible. In contrast, we adhere to the typical stream classification scenario, in which new labels are made available as the stream progresses.

In [1], Beyer et al. we propose entity-centric learning on a stream of reviews, to investigate to what extend the series of labels recorded for reviews on an entity/product are adequate to predict the label of the next review. For this investigation, we juxtapose stream classification methods that exploit the review texts with new methods that only consider the labels. Among them is a method which assigns the majority label of the entity and one which assigns the majority label of the whole stream but all of the text-ignorant methods were outperformed by the entity-ignorant and text-aware baseline. In this work, we do consider the texts and we build an ensemble, combining methods that take the relationship between observation/review and product/entity into account with methods that do not.

2.2 Classification and Forgetting on a Stream of Reviews

Concept drift is a difficult problem in most data streams [3] and a common measure to address drift is forgetting outdated data. Wagner et al. [11] proposed a Multinomial Naive Bayes (MNB) classifier that incorporates different forgetting mechanisms which fade word counts either aggressively or gradually. Fading showed better performance than an accumulative MNB when predicting tweet polarities in the presence of drift but the aggressive fading version was forgetting too fast during times of stability. In our scenario we have many short and stable entities and some entities which show drift [1]. Therefore, we employ the gradually fading MNB since it performs well on both stable and drifting data. The temporal MNB proposed in [7] is an option we would consider in future work, since this algorithm also deals with drift in the feature space.

3 OUR APPROACH

As in conventional opinion stream classification, our learning task is to predict the label of each arriving review. To exploit the fact that some reviews refer to the same entity, we partition the stream into one substream per entity, as explained in 3.1. In 3.2 we describe how model learning, adaption and forgetting are done by the entity-centric learning algorithm and by the entity ignorant one, which cooperate in an ensemble. We then present our weighting schemes for the ensemble members in 3.3.

3.1 Entity-Centric Modeling of the Stream

We model the dataset D as a stream of incoming reviews, where each review belongs to a specific product. From here on, we use the more general terms "entity" instead of product and "observation" refering to an entity/product. We denote as $t_1, t_2, \ldots, t_m, \ldots$ the timepoints of the arrivals of the observations, so that o_m stands for the observation which has arrived at timepoint t_m.

We denote the set of all entities as E, and the j^{th} observation belonging to entity $e \in E$ as $obs_{e,j} \in D$. This implies that all observations belonging to $e \in D$ constitute a substream T_e. Since the first observation for an entity may arrive at any timepoint t_m, and since the popularity of the entities varies, the substreams have different speeds, and the j^{th} observation for entity e may arrive much later than the j^{th} observation for entity e'.

Each observation consists of a text field (the review content) and the sentiment label from a set of labels $\mathcal{L}$, for example the number of stars assigned to a review, or the set {pos, neg, neutral}.

Given the infinite stream D of observations, the learning task is to build a model, which at each timepoint t_m receives an observation o_m belonging to entity $e \in E$ and predicts the label of this observation, given all reviews seen thus far for e and for all other entities.

3.2 An Ensemble with Two Voting Members

Our proposed ensemble has two voting members: a conventional stream classifier that treats all observations as independent, and the set of single-entity-classifiers (SECs), one per entity. We explain the SECs first and describe the orchestration of the ensemble thereafter.

3.2.1 The Entity-Centric Ensemble Member.

For each entity $e \in E$, we train and gradually adapt a *Single-Entity-Classifier* SEC_e. This classifier sees only the substream of observations $T_e = \{obs_{e,1}, obs_{e,2}, \ldots\}$. Since the set of entities E over the stream D is not known in advance, we perform a single initialization step for the whole D. Then, whenever a new entity e shows up, we launch a new SEC_e. A SEC_e is invoked for classification and adaption only if an observation on e arrives. In the initialization step, we build a single feature space F of size N over D, selecting the top-N words (for a very large N). When the first observation of an entity e, $obs_{e,1}$ appears, a new SEC_e is created and trained. Although we currently maintain all SECs in memory, SECs can be kept in secondary storage and retrieved only when needed, especially if the stream is slow.

As learning core for each SEC we consider a Multinomial Naive Bayes with 'gradual fading', proposed in [11]: this algorithm decays the word counts per class, depending on how long it has been since a word has been encountered for a given class. For SEC_e, we perform gradual fading on the substream T_e, wherein the count of a word per class is decayed proportionately to the last timepoint at which the word appeared in an observation of e for this class. Since for each entity e, SEC_e is trained only within the substream T_e, the conditional word counts per class, faded to different extents, differ among entities.

3.2.2 The Entity-Ignorant Ensemble Member.

The second member of our ensemble is a conventional stream classifier that ignores the entity to which each observation belongs. We denote this classifier as 'Entity IGnorant Classifier' (EIGC).

The EIGC uses the same feature space F as the SECs. Since it sees all observations of the stream D, it is initialized as soon as the first observation arrives and can be used for learning and classification thereafter. As learning core of the EIGC we use again the gradual fading MNB of [11], wherein the word counts are modified for each arriving observation and the fading of a word refers to the whole stream D, as opposed to the substream used by each SEC.

3.3 Ensemble Variants based on Weighting

We consider three weighting schemes for the ensemble members, each of them corresponding to an ensemble variant.

Variant 1: The Entity-Centric-Classifier-Ensemble. ECCE builds upon the fact that a minimum number of training observations x is necessary before a classifier can deliver reliable predictions. Hence, when the stream starts, *ECCE* initializes *EIGC* and one SEC for the entity e of each arriving observation. As soon as the minimum number of observations x has been reached for the SEC of an entity e, this classifier SEC_e can be used for predictions. Obviously, *EIGC* is the first learner to start, since it is trained on all observations. Thus, *ECCE* uses *EIGC* to deal with the cold-start problem for new entities and for rarely referenced ones. As soon as x observations have been seen for entity e, *ECCE* switches from *EIGC* to the dedicated SEC_e for observations on e. The *EIGC* is still trained in parallel so that it can benefit from the knowledge as well.

Variant 2: the Entity-Centric-Weighted-Ensemble. ECWE uses *EIGC* for the observations of some entities, even after the cold-start is over. In particular, *ECWE* assigns a weight w to the SECs of the ensemble and $1 - w$ to *EIGC*. These two weights are applied to the votes of the ensemble members for the label of each arriving observation.

Variant 3: The Entity-RMSE-Weighted-Ensemble. ERWE replaces the fixed weights used by *ECWE* with a weight emanating from the classification error of each ensemble member, thus assigning a higher voting weight to the member that has lower error. In particular, for each arriving observation o, let e be the entity to which o belongs. We define the weight assigned to SEC_e as

$$wSEC(e) = \frac{RMSE(EIGC_e)}{RMSE(EIGC_e) + RMSE(SEC_e)}$$

The weight assigned to *EIGC* for that entity e is

$$wEIGC(e) = \frac{RMSE(SEC_e)}{RMSE(EIGC_e) + RMSE(SEC_e)}$$

where we use the Root Mean Square Error (RMSE) as misclassification error, assuming ordinal labels. If the labels have no internal order, as would be the case for positive and negative labels only, the misclassification error can be used instead of RMSE.

Note that the weight of the *EIGC* vote for an observation depends on the entity to which this observation belongs. This allows the ensemble variant *ERWE* to assign higher weights to *EIGC* on entities, for which the SEC does not perform well (yet), while giving preference to the SECs, as soon as they show superior performance. Furthermore, this method is parameter free in contrast to the *ECWE* where we have to pick the weights in advance.

4 ENTITY-CENTRIC EVALUATION SCHEME

For the evaluation of conventional stream classification algorithms, it is typical to specify a performance metric, e.g. misclassification error, $\kappa+$ or RMSE, and then compute it over the stream's lifetime, typically using prequential evaluation [12]. We must rather evaluate to what extent entity-centric learning is beneficial when predicting the labels of the observations belonging to each entity. In particular, we study following aspects:

- For which entities does entity-centric stream learning lead to better classification of the opinionated texts than conventional opinion stream classification?
- How does the initialization phase, i.e. the threshold on the number of observations seen after cold-start, affect the ensemble's performance?

To this purpose, we introduce an entity-centric evaluation framework. Additionally to this proposed method we also report the performance in a traditional way where we show the *RMSE* of the

different methods on non-overlapping chunks, with each chunk containing 1000 predictions. We use *RMSE* as a performance measure because we have ordinal labels which implies that we do not only want to know if our prediction is correct but also how far off was our prediction in case we were wrong, e.g. it is better to predict a 5-star review as being 4 stars than predicting it to be 1 star.

Our entity-centric evaluation framework consists of three components.

First component: the role of the threshold on the number of observations in the initialization phase. We first align the substreams of the entities and create a vector of the next n predictions of each entity for a given threshold x, where x is the amount of training data that the *SECs* have been trained on. The prediction vector is called V_P and the vector of the true labels is called V_T. We can now calculate the *RMSE* between those two vectors, e.g. $RMSE(V_{ECCE}, V_T)$. We compare the ensemble based *RMSEs* against our baseline which is $RMSE(V_{EIGC}, V_T)$. Figure 1 depicts this part of the workflow by means of an example when we follow the first arrow to the right and then the first arrow downward.

Second component: error reduction at entity level. To find out how many entities do actually perform better compared to our baseline, we calculate the *RMSE* on the next n predictions per entity which is depicted in the table to the right in figure 1. If the *RMSE* of an ensemble-based prediction is better than the *RMSE* of the baseline prediction for a given entity, we consider it to be a *win* of the ensemble-based method. This procedure enables us to calculate the percentage of entities for which the ensemble won against the baseline.

Third component: Statistical testing across the time axis. We also test for each x whether the performance of the ensemble is significantly better on an entity-level compared to the baseline with a Wilcoxon signed rank test. The procedure is as follows:

- Filter out all predictions for which we do not have at least $n = 100$ predictions at x.
- For each entity collect the next $n = 100$ predictions.
- For each entity calculate the $RMSE_{Ensemble}$ and the $RMSE_{EIGC}$. This returns two vectors each of the size of the number of entities which still contribute predictions.
- Perform a Wilcoxon Signed Rank Test on the vectors from the previous step.

5 EXPERIMENTS AND RESULTS

Goal of our experiments is to investigate to what extent entity-centric learning improves classification over a stream of opinionated documents. For the evaluation, we compare against the entity-ignorant classifier *EIGC*, and use the evaluation framework of the previous section. Since our objective is to identify performance improvements, if any, we use only the gradually fading MNB of [11] as classification algorithm and consider the simple vectorization scheme on ranking of words employed also in [1].

5.1 Datasets and Parameters

We use following two subsets of the Amazon dataset presented in [4]:

- 'Tools and Home Improvement', denoted as **tools** hereafter, contains 260,659 entities (products) and 1,926,047 reviews on them. We removed all entities with less than 10 reviews because of memory constraints, retaining 33,989 entities and 1,416,766 reviews.
- Watches', denotes as **watches** hereafter, contains 78,162 entities and 487,741 reviews on them.

Each review consists of a review text and a star-rating ranging from 1 to 5 stars. The star-rating is the label that we want to predict based on the review text and is of ordinal nature.

We use the following parameters in our experiments:

- We use the top $N = 1000$ words for vectorizing the reviews.
- We start our plots at threshold $x = 2$ and incrementally increase it until the number of entities which are part of the evaluation vector drops below 5, as statistical tests with less than 5 samples become unreliable according to [6]
- The number of predictions each entity has to contribute at a given x is $n = 100$.
- The weights for the *ECWE* are $w = 0.2, 0.5, 0.8$. As there was not a big difference in the performance we will only report on the best results which were achieved with $w = 0.5$.
- The chunk size for the non-overlapping *RMSE* evaluation is set to 1000.

5.2 Evaluation Procedure

We follow the evaluation framework described in section 4. For each stream classifier $Cl \in \{ECCE, ECWE, ERWE, EIGC\}$ we compute the $RMSE_{Cl}$ after seeing at least $x >= 2$ observations for each entity. The *EIGC* can start performing predictions earlier, but since our objective is to study the contribution of the entity-centric models on performance, we can only compute the performance after the *SEC*s can start making predictions. For the RMSE computations, we ignore the first prediction of each classifier, since this prediction is not informed through previous data. We thereby vary the threshold x that captures the number of observations per entity seen by the single entity-centric classifiers, the *SECs*, of the ensembles *ECCE*, *ECWE* and *ERWE*.

To better understand how the ensembles affect performance, we further count the entities which benefit from the entity-centric classification, as well as the entities removed as x is increased. It must be noted that although the RMSE values are computed on the same number of observations for *ECWE*, *ECCE*, *ERWE* and *EIGC*,the *EIGC* exploits more observations than the *ECCE* at each timepoint. The reason is that observations are incorporated into the *EIGC*, as soon as learning starts, i.e. earlier than for the *SECs*. Hence, it is expected that *EIGC* will show lower RMSE than our approach, at least for small values of x.

5.3 RMSE and Number of Entities

We first evaluate on conventional *RMSE*, considering non-overlapping chunks. The performance results are shown on Figure 2 for **tools** (left part) and **watches** (right part). For both datasets, *ECCE* performed worst and *ERWE* performed best while the *ECWE* is better than *EIGC* on **tools** but slightly worse on **watches**.

Conventional RMSE does not capture the contribution of each entity to each chunk. This contribution is captured on Figure 3 for the **tools** dataset. We organize the entities on the number of observations for which predictions are made, n, and specify 5 bins:

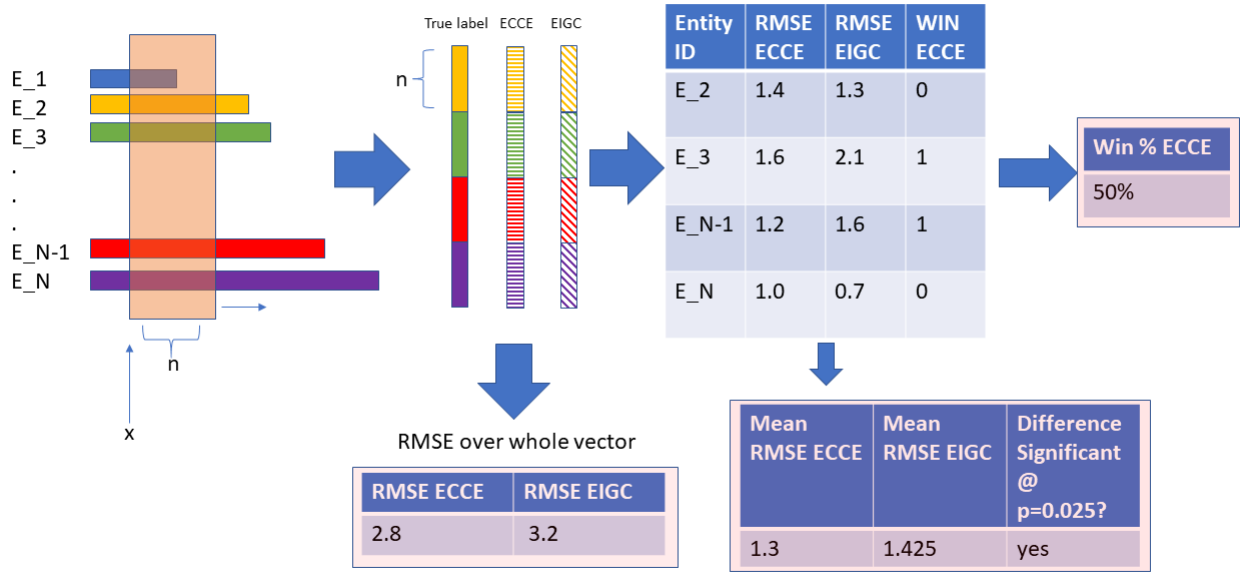

Figure 1: The figure shows the entity-centric evaluation using *ECCE* as an example. On the left we can see how the entities are aligned and how the next n observations are extracted at a given threshold x. Entity E_1 in blue does not qualify because it is too short and cannot contribute n observations at this x. Following the arrow to the right we can see how the n observations of each entity now form the vector of true labels and the predictions of the *ECCE* (dashed lines) and *EIGC* (diagonally dashed lines). Following the downward arrow we get the result of the first component which is the *RMSE* at a given threshold x. If we follow the arrow to the right, we see the *RMSE* per entity. This table is used to create the results for the second component to the very right, the percentage of wins, and the third component at the bottom, the test for statistical significance.

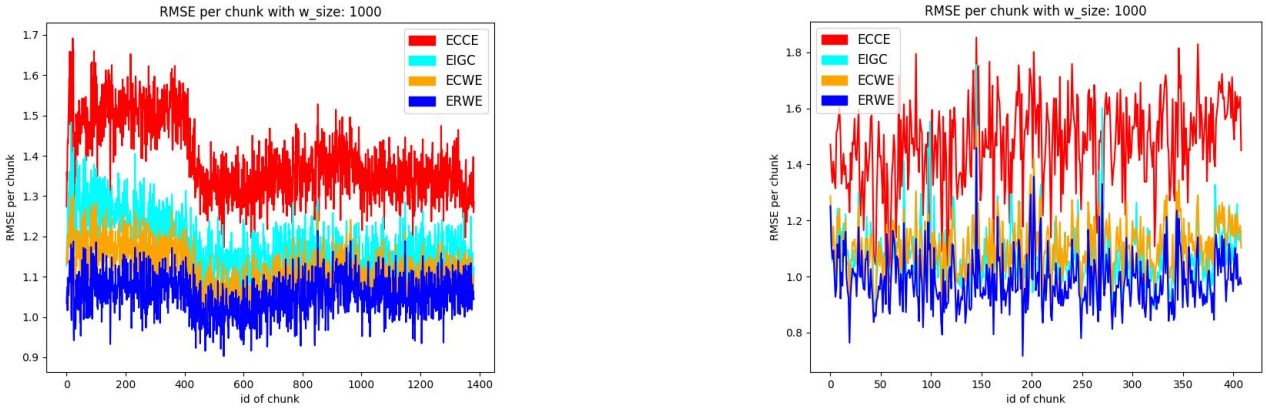

Figure 2: Datasets tools (left), watches (right): *RMSE* on non-overlapping chunks (chunk size=1000 predictions); lower values are better

the first bin accommodates entities with n of no more than 24, while the 5th bin accommodates entities that contribute to more than 264 predictions. In each bin, we show the percentage of error as distance (in number of stars) between the true label and the predicted one: the legend at the rightmost part of the figure shows the colors as the error increases from zero to 4 stars.

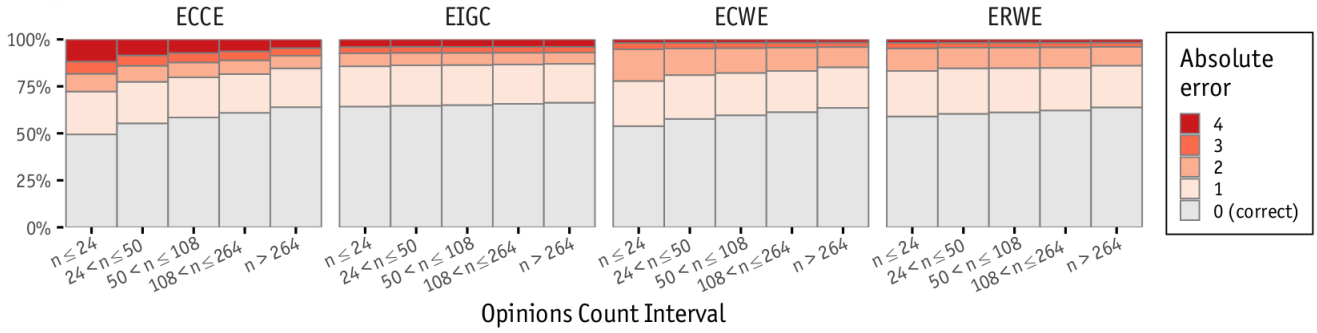

Figure 3: Dataset tools: "Absolute error" as absolute difference between number of predicted and true stars for a review, plotted against the number of predictions contributed by each entity - organized in 5 bins; smaller percentages of error are better, especially for the most severe error type of 4 stars

As we see at the leftmost part of Figure 3, the *ECCE* has lowest performance for the bins containing low values of n and comparable performance for entities that contribute with many ($n > 264$) predictions. This also explains why *ECWE* and *ERWE* have fewer correct predictions than *EIGC*. On the other hand, the *ECCE* seems to prevent *ECWE* and *ERWE* from large errors (3 or 4 stars apart the true label); large errors occur slightly more often under *EIGC* than under *ECWE* and *ERWE*.

We next measure the impact of entity-centric learning with the evaluation framework of the previous section. We vary the threshold x on the number of observations and show the entity-centric *RMSE* (Figure 4: **tools**, Figure 6: **watches**), the percentage of entities for which entity-centric learning leads to better predictions than *EIGC* does (upper part of Figure 5: **tools** and of Figure 7: **watches**) and absolute number of entities (lower part of Figures 5 and 7).

On Figure 4, we see for **tools** that the RMSE curve of *ECCE* crosses that of *EIGC* at around $x = 500$ and shows improvement thereafter, while the best performance is achieved by *ERWE*. On Figure 6, we see that the performance curves on **watches** are similar, except that the RMSE curve for *ECCE* starts crossing the curve of *EIGC* at $x = 400$.

In the upper part of Figure 5 for **tools**, we see that the majority of entities benefit from *ECCE* at some point ($x > 700$). This point is higher than for the absolute *RMSE* ($x > 500$). *ECWE* and *ERWE* show wins over *EIGC* for 50 percent and more of the cases. The plots on the upper part of Figure 7 for **watches** show that the majority of entities rarely benefit from *ECCE*, while *ERWE* does improve the *RMSE* for most entities.

In the lower part of Figures 5 and 7 we see that the number of entities contributing to the predictions drops rapidly as we increase x. The figures indicate that the use of entity-centric models can increase the performance with regards to *RMSE*, but only after acquiring a minimum number of observations, especially if we use the *ERWE* which shows lower entity-centric *RMSE* in both datasets. There seems to be a cutoff point around $x = 500$ where the *RMSE* curves of the *ECCE* and *EIGC* cross and from which on the *ECCE* seems to perform better. The main difference betweens the datasets is that *RMSE* curves of the *ECCE* and the *EIGC* are much closer on **watches**. In order to find the exact point at which a majority of the entities starts to benefit from the entity-centric models we conducted significance tests at certain x which will be discussed next.

5.4 Significance Testing

We used scikit-learn [9] for the significance tests. We compare *ECCE* (the weakest ensemble) and *ERWE* (the ensemble with the mostly best performance) to *EIGC*. For each x, we apply Wilcoxon signed-rank test in order to find out if the observed difference in *RMSE* is significant at a $p = 0.025$ confidence level as described in the third component in section 4. The results are on Figure 8 for **tools** (upper part) and **watches** (lower part). For each value of x, we assign an 1 if the entity-centric model is significantly better, a -1 if it is significantly worse and a 0 if there is no significant difference.

For **tools** we see in the upper part of Figure 8 that *ECCE* is inferior to *EIGC* until $x = 490$, performs similarly to *EIGC* until $x = 1170$ and occasionally shows superior performance for higher values of x. In contrast, *ERWE* is almost always significantly better than the *EIGC* except for very high values of x. This also agrees with the percentage of wins shown on Figure 5.

For **watches** we see in the lower part of Figure 8 that *ECCE* is inferior to *EIGC* until $x = 350$ and from then on both models perform similarly. *ECCE* never becomes superior to *EIGC*, perhaps because **watches** contains only very few entities with many observations: at $x = 600$ there are only 20 entities left. As for **tools**, *ERWE* is mostly significantly better than *EIGC* except for large x.

5.5 Overhead of the Entity-Centric Ensembles

In this last experiment, we compare the memory usage (more precisely: heap usage) and execution time of *EIGC* to that of the entity-centric ensembles. On Figure 9, we show how these quantities change with the number of entities for the larger of the two datasets, namely **tools**. Since we are not interested in prediction quality here, we ignore the predictions, whereupon *ECCE*, *ERWE* and *ECWE* behave identically; we denote them jointly as "entity-centric" (ensemble). We can see that there no differences in execution time but the memory usage increases for the entity-centric ensemble. However, the increase is very slow. The increase depends on the number of entities in a given dataset. For **tools**, the mean memory usage

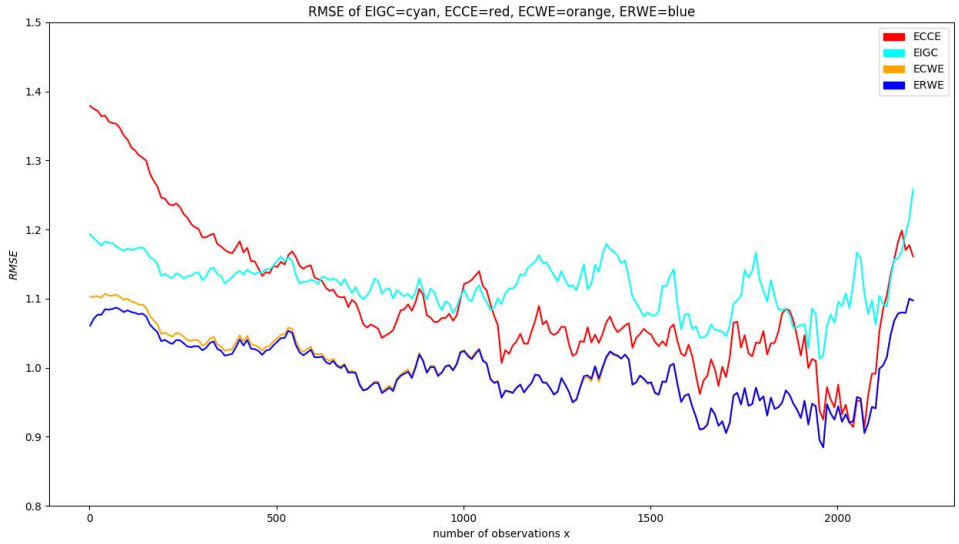

Figure 4: Dataset tools: Entity-centric *RMSE*; lower values are better

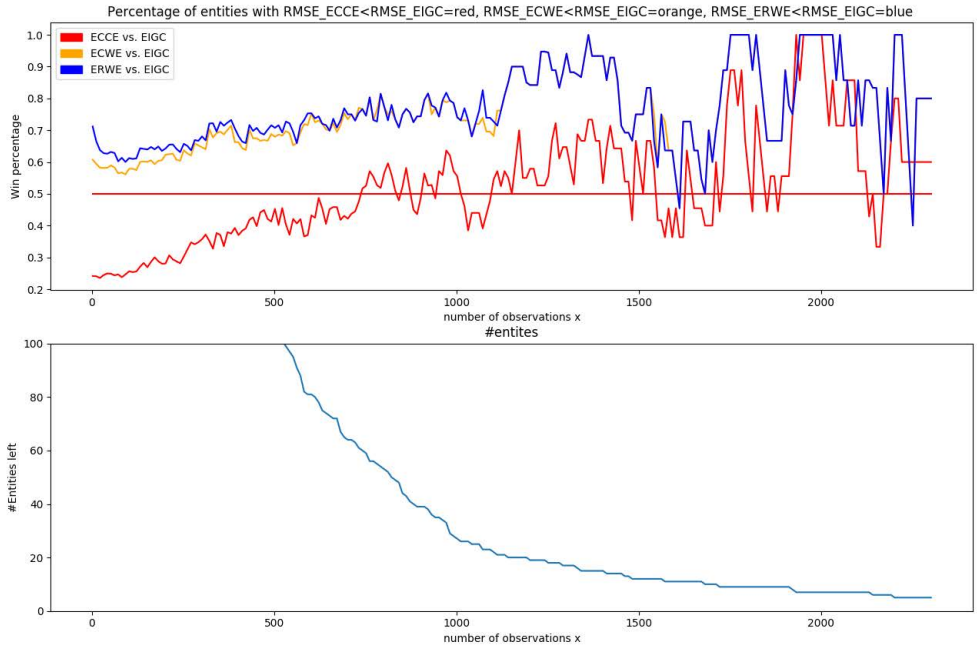

Figure 5: Dataset tools - Top: Percentage of entities at x for which the *RMSE* of an entity-centric learner is better than the *RMSE* of *EIGC*: *ECCE* vs *EIGC* (red); *ECWE* vs *EIGC* (orange) and *ERWE* vs *EIGC* (blue); higher values are better.
Bottom: Number of entities still retained at x.

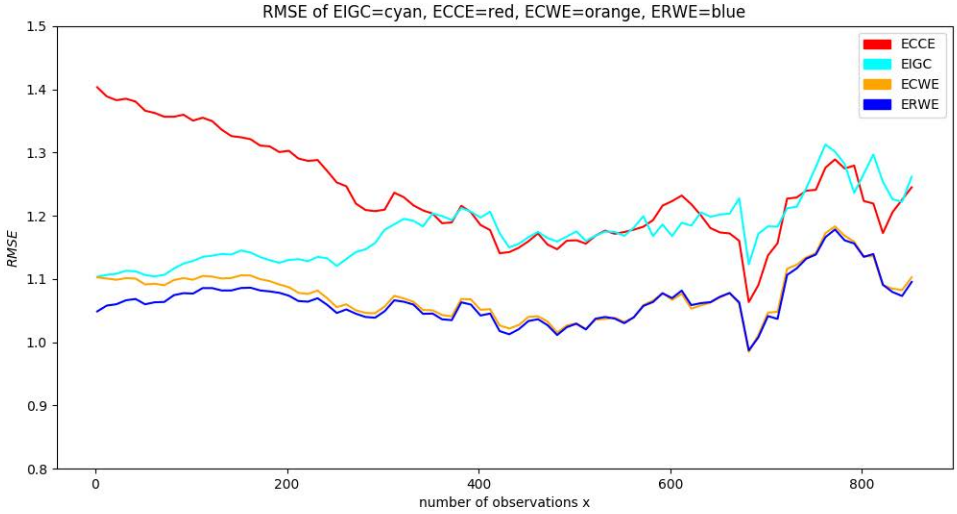

Figure 6: Dataset watches: Entity-centric *RMSE*; lower values are better

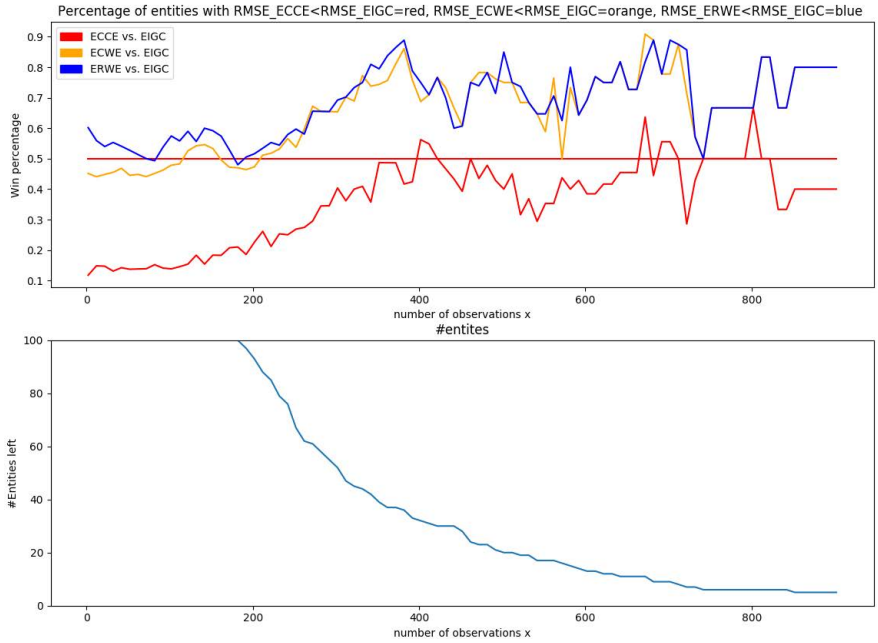

Figure 7: Dataset watches - Top: Percentage of entities at x for which the *RMSE* of an entity-centric learner is better than the *RMSE* of *EIGC*: *ECCE* vs *EIGC* (red); *ECWE* vs *EIGC* (orange) and *ERWE* vs *EIGC* (blue); higher values are better.
Bottom: Number of entities left at x.

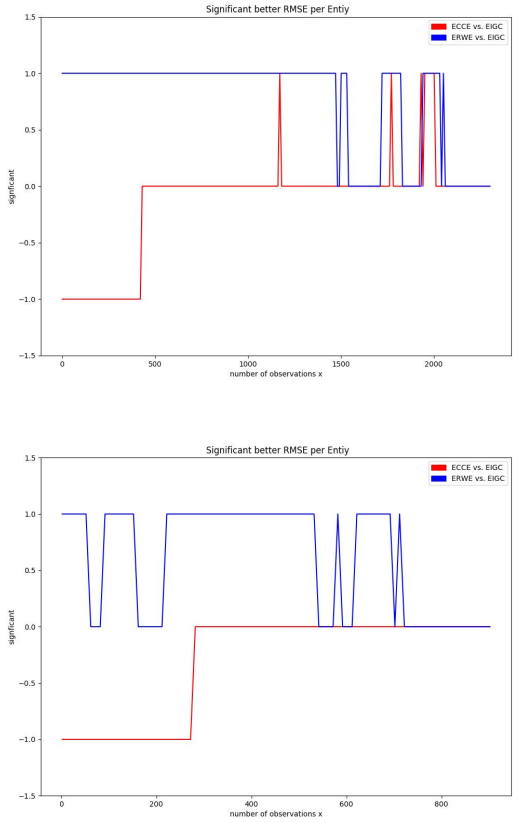

Figure 8: Datasets tools (top), watches (bottom): Plot of significance testing with Wilcoxon signed-rank test ($p = 0.025$), juxtaposing the *RMSE* of *EIGC* to that of *ECCE*, resp. of *ERWE*; an 1 stands for significantly higher performance of an entity-centric ensemble in comparison to *EIGC*, and -1 stands for the opposite; *ERWE* is the winner for most values of x

of the models were almost 6GB (max 9.3GB) for the entity-centric ensemble and 2GB (max 2.9GB) for the entity-ignorant method so in this case the entity-centric ensemble used between two and three times the memory of an entity-ignorant approach.

6 CONCLUSION AND OUTLOOK

In this work we investigated the usefulness of entity-centric stream learning. We devised an ensemble that combines an entity-ignorant learner over the whole data stream with a set of single entity learners, each one seeing only the substream of that entity.

6.1 Reflection on our Findings

Our results show that entity-centric learning can improve classification performance in a data stream, especially when using voting with weights – the ensemble *ERWE*. While *ERWE* showed consistently high performance on both datasets, the simple ensemble

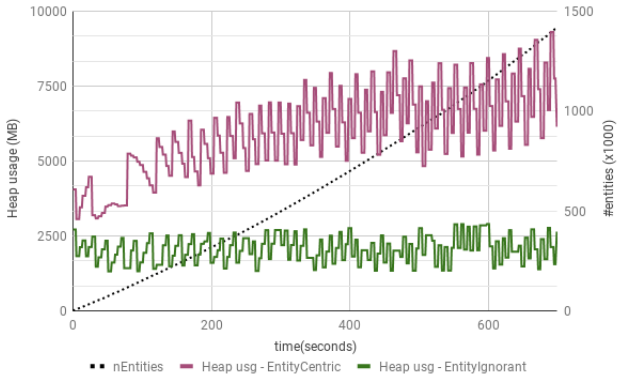

Figure 9: Dataset tools: Memory usage and computational time of the entity-centric ensemble vs *EIGC*, as well as number of entities seen thus far (dotted line): entity-centric learning has only modestly higher memory demand, while the execution time is comparable to that of *EIGC*

ECCE rarely outperformed the entity-ignorant baseline. This indicates that the participation of an entity-centric learner contributes to performance quality, but only if the votes are weighted on the basis of past performance. It is stressed that the superiority of the voting scheme *ERWE* holds even on entities that have few observations.

Our comparative study required an evaluation framework that went beyond conventional RMSE on a stream. The new framework required entity alignment, thus loosing the notion of absolute time, as well as a training phase, thus excluding entities with very few observations. For example, in our experiments, we required at least 5 entities with at least $n = 100$ observations per entity. On the other hand, the evaluation framework shows how many entities acquire better predictions from entity-centric learning, thus delivering insights on how beneficial entity-centric learning is for a given dataset. Hence, our evaluation framework is appropriate for deciding whether entity-centric learning should be applied on a given dataset: once this is decided, the simpler RMSE on non-overlapping chunks can be used.

6.2 Future Work

Our first results on entity-centric learning are encouraging, yet several further steps are now due. First of all, the evaluation framework prevented the study of entities with very few observations. Hence, we intend to consider further evaluation methods that are less data demanding.

Storing a learner for each entity observed seems impractical. Therefore, we face following questions for future work: (i) How can we decrease the number of *SECs* in order to save storage? (ii) When does it make sense to terminate a *SEC*, i.e. ignore the corresponding entity during learning? (iii) How do we decide when to start a *SEC*? For the first question, entity clustering seems appropriate. We will consider the method of [10] as starting point: Unnikrishnan et al. use the static properties of entities to find an entity's nearest neighbours and learn from them. For the second question, we anticipate ways of

monitoring the elapsed time since the last observation for an entity. For the third question, the number of observations of an entity within a time frame or window may be a proper heuristic to begin with. We also intend to investigate the potential of multi-armed bandits for deciding whether to invoke *EIGC* or a *SEC* at each timepoint, notwithstanding that an arm's reward may be difficult to compute if there are only very few observations for an entity and given the presence of drift.

Our current investigation has focused on opinionated documents with ratings. We want to consider more elaborate vectorization methods for text data, but also different types of datasets, outside the text classification domain. Additionally, we plan to consider different stream classification algorithms, next to the currently used stream version of Multinomial Naive Bayes.

7 ACKNOWLEDGMENTS

This work is partially funded by the German Research Foundation, project OSCAR "Opinion Stream Classification with Ensembles and Active Learners". Additionally, the first author is also partially funded by a PhD grant from the federal state of Saxony-Anhalt.

REFERENCES

[1] Christian Beyer, Uli Niemann, Vishnu Unnikrishnan, Eirini Ntoutsi, and Myra Spiliopoulou. 2018. Predicting Polarities of Entity-Centered Documents without Reading their Contents. In *Proceedings of the Symposium on Applied Computing*. ACM.

[2] Pavlos Fafalios, Vasileios Iosifidis, Kostas Stefanidis, and Eirini Ntoutsi. 2017. Multi-aspect Entity-centric Analysis of Big Social Media Archives. In *International Conference on Theory and Practice of Digital Libraries*. Springer, 261–273.

[3] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 46, 4 (2014).

[4] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*. 507–517.

[5] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. 2009. Context-aware systems: A literature review and classification. *Expert Systems with applications* 36, 4 (2009), 8509–8522.

[6] Richard Lowry. 2014. Concepts and applications of inferential statistics. (2014).

[7] Damianos P. Melidis, Myra Spiliopoulou, and Eirini Ntoutsi. 2018. Learning under Feature Drifs in Textual Streams. In *Proceedings of the 2018 ACM on Conference on Information and Knowledge Management*. ACM. to appear.

[8] Kenta Oku, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. 2006. Context-aware SVM for context-dependent information recommendation. In *Proceedings of the 7th international Conference on Mobile Data Management*. IEEE Computer Society, 109.

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

[10] Vishnu Unnikrishnan, Christian Beyer, Pawel Matuszyk, Uli Niemann, Ruediger Pryss, Winfried Schlee, Eirini Ntoutsi, and Myra Spiliopoulou. 2018. Entity-Level Stream Classification: Exploiting Entity Similarity to Label the Future Observations Referring to an Entity. In *Data Science and Advanced Analytics (DSAA), 2018 IEEE International Conference on*. IEEE. to appear.

[11] Sebastian Wagner, Max Zimmermann, Eirini Ntoutsi, and Myra Spiliopoulou. 2015. Ageing-based Multinomial Naive Bayes Classifiers over Opinionated Data Streams. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECMLPKDD'15*, Vol. 9284. 401–416.

[12] Indrė Žliobaitė, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. 2015. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning* 98, 3 (2015), 455–482.