Multi-domain and Explainable Prediction of Changes in Web Vocabularies

Albert Meroño-Peñuela
Department of Informatics
King's College London
United Kingdom
albert.merono@kcl.ac.uk

Romana Pernisch
Department of Computer Science
Vrije Universiteit Amsterdam
Netherlands
r.pernisch@vu.nl

Christophe Guéret
Accenture Labs
Dublin, Ireland
christophe.gueret@accenture.com

Stefan Schlobach
Department of Computer Science
Vrije Universiteit Amsterdam
Netherlands
k.s.schlobach@vu.nll

ABSTRACT

Web vocabularies (WV) have become a fundamental tool for structuring Web data: over 10 million sites use some structured data format to markup content, and an ever-growing number of ontologies are used to specify types and relations in billions of Knowledge Graph statements. Maintaining these vocabularies, and keeping up with their changes, are manual tasks with very limited automated support. Publishers invest significant labour in updating them to fit new content; while users need the latest version to update their data markup. Recent work shows that machine learning can be used to reliably predict vocabulary changes, but these depend on the specific application domain (e.g. biomedicine), and do not explain what aspects of changes (e.g. their type, frequency, etc.) have an impact in their prediction. In this paper, we describe a framework that uses multiple supervised learning models to learn and predict changes in versioned vocabularies, independent of their domain. Using well-established results in ontology evolution we extract domain-agnostic and human-interpretable features and explain their influence on change predictability. Applying our method on 139 WV from 9 different domains, we find that ontology structural and instance data, the number of versions, and the release frequency highly correlate with predictability of change. These results can pave the way towards integrating predictive models into knowledge engineering practices and methods, supporting publishers and consumers in proactively versioning vocabularies and updating datasets.

1 INTRODUCTION

Increasingly, the Web contains more and more structured data describing people, organizations, locations, and products, using standards such as RDF, Microdata, JSON-LD, RDFa, and Microformats [22]. Structured vocabularies [12], SKOS taxonomies [21] and OWL ontologies [19] play a crucial role in this process as they provide terminologies to describe such domains in Knowledge Graphs, formalizing the semantics of multiple domains, and extending interoperability. Concepts are central entities in these vocabularies, and represent objects with common characteristics. However, as more and more data populate the Web and its services, these concepts are continuously subject to change. Consequently, Web vocabularies (WV) need to change in new versions and adapt to the new realities. Schema.org [12] and DBpedia [18], e.g., model cross-domain knowledge and are updated every 1-2 months[1]. The Historical International Standard Classification of Occupations (HISCO) [17] is a taxonomy of historical occupations since the 16th century permanently accepts additions from its users. The Gene Ontology (GO) [6] standardizes the representation of gene attributes across species and datasets and publishes new releases monthly. These updates are generally a manual, unassisted, and knowledge intensive task. To adapt their vocabularies to domain changes, data publishers typically use their expert knowledge to produce new *vocabulary versions*. This creates *vocabulary version chains*: subsequent unique states of a vocabulary with unique identifiers. These version chains exacerbate the manual vocabulary management practices of publishers, who face hard questions on what parts of their vocabulary needs attention; and users, who are faced with questions on what version to use and whether the wrong version will make them loose features. The automatic detection of concept change and shift in meaning would be a great aid in proactively supporting vocabulary publishers and users in these challenges.

In previous work, Pesquita and Couto [26] show that feature engineering [29] can be effective for predicting class *enrichment* of biomedical *OBO/OWL* ontologies. González and Hogan [11] introduce a new algebra over characteristic sets, and use it in eleven weeks of Wikidata snapshots to predict future changes. However, these methods have two important pitfalls. First, they have been evaluated in only one application domain; questions about their generality and domain independence remain open. Second, they provide no explanations for ensuring good predictability of vocabulary change; e.g., how often versions need to be released, what types of changes have a deeper impact in change predictability, etc. Therefore, our interest is to investigate a more generic approach for predicting *when* and *where* a Web vocabulary of any domain will change; and to explain what specific features make vocabulary changes more predictable. We build on ([26, 29, 34]) and propose a *generic vocabulary change prediction framework* based on feature extraction and supervised learning on past versions of Linked Datasets. This framework predicts *change* in arbitrary schemas (vocabularies, taxonomies, ontologies) in RDF graphs in a domain-independent manner; and offers human-understandable *explanations* for such changes through human-interpretable features and feature ranking algorithms. Our research questions are:

RQ1. To what extent can we use past versions to *predict concept change* in WV independently of the domain of application?
RQ2. What *per-version features* in past vocabulary versions have a greater influence on predicting concept change in WV?
RQ3. What *per-chain features* in past vocabulary versions have a greater influence on predicting concept change in WV?

[1] https://schema.org/docs/releases.html

In order to address these, we apply our proposed framework to 139 different Web vocabulary version chains in RDF, including the Dutch historical censuses [20], the DBpedia ontology [18], vocabularies from SPARQL endpoints in the Linked Data cloud [33], and Linked Open Vocabularies used all over the Web. We obtain solid evaluation performances, with F-measures of 0.84, 0.93 and 0.79 on test data. We explain the datasets in which our approach works best, and find that features such as dataset size, the number of versions in the chain, the time gap between each version or the tree-depth of their hierarchies have an influence in the quality of the predictive models. Therefore, our contributions are:

- A domain-independent and reusable framework for learning and predicting concept change in vocabulary version chains; based on combining, integrating, and leveraging various existing machine learning building blocks.
- To the best of our knowledge, the largest and most comprehensive collection of features explaining vocabulary change on a per-version and per-chain vocabulary basis, along with tools to compute them. We show that human-engineered features combined with feature selection [15] are very effective in addressing explainability of vocabulary change models learned from data.
- A large scale, multi-domain evaluation of the performance at learning and predicting ontology change in 139 versioned ontologies from 9 different domains (e.g. academia, government/organisations, geographic, cultural, etc.).

The rest of the paper is structured as follows. In Section 2, we survey previous efforts addressing the problem of vocabulary change. Section 3 describes our approach, pipeline and feature set. In Section 4 we present our evaluation, describing our datasets and experimental setting. In Section 5 we show our results, and discuss them with respect to our research questions, before we conclude in Section 6.

2 RELATED WORK

In Machine Learning, changes in the domain are related with the phenomenon of concept drift [9], which occurs when "the concept of interest may depend on some hidden context, not given explicitly in the form of predictive features. (...) Changes in the hidden context can induce more or less radical changes in the target concept, which is generally known as concept drift" [31]. Multiple concept drift detection methods exist [9]. Similarly for ontologies, SemaDrift is a framework for calculating semantic drift in ontologies [28]. The authors define various related terms such as semantic drift, change and decay but also concept change and drift, a "transformation of meaning of ontology's underlying concepts between versions" [28]. Further, OntoDrift [4] extends the ideas and implementation of SemaDrift, by considering more aspects of a context. They also account for additions of new concepts and removal of old ones, by adding the Jaccard Index to their drift measure [4]. Although effective for computing pair-wise drifts, SemaDrift and OntoDrift are heuristic-based and therefore do not scale well to large datasets.

In the Semantic Web, changes in concepts can be studied through formal differences between ontologies in Description Logics [10]. [7] propose a method based on clustering similar instances to detect concept change. [13] focus on semantic drift definition based on concept signatures within ontologies. The related field of ontology evolution deals with "the timely adaptation of an ontology and consistent propagation of changes to dependent artifacts" [11]. Accordingly, change is only a step in the evolution process, although the definition of the goal of ontology change ("deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements") [13, 16] suggests that the overlap between the two fields is considerable.

Predicting changes can also be seen as ontology forecasting, i.e. predicting which new concepts are going to be added to the ontology by only using past knowledge. For this purpose, [3] introduce the Scientific Innovation Forecast (SIF) model. Their approach outperforms known baselines when forecasting over 5 years. The field of graph completion and link prediction is also related [22]. However, we do not aim at predicting new concepts or links, but rather to predict where changes are going to occur within an ontology, to already present concepts. Gonzalez et al. [11] define an algebra using characteristic sets, which they use on a change learning and prediction task. They apply it to eleven weeks of data from Wikidata, by converting it to characteristic sets and using diffs between versions to predict the following changes. However, they do not use version chains and the additional step of conversion makes it hardly comparable to versioned, human-engineered ontologies. [24] introduce a new strategy for updating RDF links by predicting triple-level changes, and using this predictions to identify what RDF documents to update. However, their focus is not on the change prediction but rather on the update which follows. Robust learning algorithms in ontology streams with semantic drift have also been investigated [5, 25]; however, graph streams demand a much higher update frequency than the classic ontology versions we study here.

In the closest work to ours [25], authors propose a method based on supervised learning on past ontology versions to predict enrichment of classes of biomedical ontologies. Models of change are learned from data, and features are engineered according to the guidelines described in [29]. [29] shows evidence that good vocabulary change predictors are generally related to (a) the structure of classes, subclasses and properties in an ontology; (b) the instances belonging to them; and (c) the usage of classes, properties and instances in applications. Together with the definitions of concept change described in [34], [26] and [29] provide an ideal framework for studying the effectiveness of supervised learning for Web vocabulary change. On the one hand, these methods have not been applied in various application domains, questioning their generality; and provide no explanations nor recommendations for good prediction results, e.g., how often versions need to be released, what changes have deeper impacts, etc. This paper addresses these pitfalls.

3 APPROACH

Our proposal builds on previous work in ontology evolution [29], supervised learning for ontology extension prediction [26], and concept drift detection [34]. We propose a supervised learning framework that generalizes the approach in [26], empirically extends and concretizes the features in [29], and automatically labels vocabulary changes according to [34]. In [34], vocabulary changes are defined

by considering the intension (i.e. de nition of classes and relations), extension (i.e. instances of those classes and relations) and labels (i.e. their identifiers) of concepts/classes. The intension of a concept is the disjoint union of a rigid and a non-rigid set of properties $\iota(8,C,1) \cup \iota(8,C,2)$. Identity over time is addressed through rigid intension equivalency, i.e. $\iota(8,C,1) = \iota(8',C,1) \Leftrightarrow \iota(8,C,2) = \iota(8',C,2)$. Intension, extension, and label similarity functions $\sigma_{int}(8,8') \in [0,1]$, $\sigma_{ext}(8,8') \in [0,1]$, $\sigma_{lab}(8,8') \in [0,1]$ quantify meaning similarity between identical concepts in different versions of a vocabulary.

The task of change prediction can be defined as follows: given an ordered set, or sequence, of versions of a vocabulary V (with or without instance data), where each version is an updated copy of a previous version with a new timestamp. Based on the notion of identity as defined above, the task of change prediction is to decide for a concept in the most current version whether is a candidate to be changed or not, and if so with what type of change (intensional, extensional or label change as defined in [34]). The notion of a "candidate to be changed" is a soft notion, mimicking the expected behaviour of a Knowledge Engineer. In our experiments we will use previous modelling behaviour to evaluate the quality of our prediction with respect to unseen, future changes.

More specifically, our framework includes: (a) an abstraction of the input parameters required for the learning process; (b) an abstraction of features that apply not only to ontologies, but to other Linked Datasets such as vocabularies and taxonomies; and (c) a pre-learning optimization technique to merge features of identical concepts between versions, into single training/test individuals.

3.1 Framework

Figure 1 shows the pipeline of our proposed framework. Taking input of Feature generation parameters, change definition, version chain, learning parameters, the system returns output of Feature selection, classifier performance.

First, the Feature Generator (FG) generates training datasets and one test dataset, according to the following input set elements: (a) version chain containing $\#$ versions of a vocabulary, in any RDF serialization, where the change prediction is to be performed; (b) several user-set feature generation parameters that control the feature generation process (the EC parameter, setting the version to be used to decide if a concept of the training dataset has changed; and the TT parameter, setting the version to be used to decide if a concept of the test dataset has changed); and (c) a customizable definition of change that determines the value of the target variable.

In our approach, we use the framework in [34] to automatically estimate change labels, using an ensemble of intension, extension, and identifier changes. The last element of the input set, learning parameters, is passed further to be used in a later stage. Once all set, training datasets and the test dataset are built by the FG as shown in Figure 2. The parameters $\#$, and)) are used to determine which versions will play the role of $\{+_C\}$, $+_A$ and $+_4$. $\{+_C\}$ is the set of training versions, which are used to build the training dataset. $+_A$ is the reference version, against which all versions in $\{+_C\}$ are compared, using the definition of change provided as input, to determine whether there is concept change or not. $+_4$ is the evaluation version and is used to build the test dataset, following a similar procedure as with $\{+_C\}$ and $+_A$, this time comparing $+_A$ with $+_4$. $+_4$ is set by default to the most recent version. While extracting features, each concept is labelled depending on whether change is detected between one version of the concept and the next, according to the definitions of [34].

Since versions can only be compared pairwise, the FG produces training datasets. In order to preserve identity of learning instances, the Identity Aggregator (IA) matches concepts in the training datasets and merges their features into one individual, modifying the dataset dimensionality accordingly. We use a simple instance-based matching, based on string similarity of resource identifiers (URIs) with a tolerance to namespace changes and minimal typing errors. The training and test datasets are then passed through the Normalizer module (Norm), which adjusts value ranges, performs necessary encoding to feature names and types by discarding outliers.

Then, we feed them into the Machine Learning Interface (MLI) for the feature selection and classification tasks. Here, we build on top of the machine learning algorithms and models provided by the WEKA API [14], indicated by the dashed arrow in g:pipeline. The last element of the pipeline's input set, learning parameters, allows for domain-independent customization and contains: (a) a feature selection algorithm (FS: Relief [15]) to rank features; (b) a relevance threshold (C) to filter selected features; and (c) the list of classifiers to be trained. The MLI runs the chosen feature selection algorithm, it trains the chosen subset of machine learning classifiers, and it evaluates the trained models.

3.2 Feature Set

We use two types of features, identifying locally and globally important features according to the literature [29] and experimental datasets (Section 4.1). The first set are per-version features. These features are calculated for every specific version in a version chain, yielding multiple values, depending on how many versions there are. The second are per-chain features, which characterise an entire chain, not each version separately. They focus on changes and average characteristics of the vocabularies. Table 1 summarizes all the features in our approach. We propose sets of conceptual features and membership features. Structural features measure the location and the surrounding context of a concept in the schema, such as children, siblings, depth of a concept (i.e. distance to the leaves), etc. Since WV are graphs in general and may contain cycles, these properties are defined with a $maxG$ threshold that indicates the maximum level at which the property will be calculated (e.g. direct children, children at depth one, two, etc.).

4 EVALUATION

The source code implementing our proposed approach in Figure 1 is available online.[2] We apply our approach on 139 different vocabularies, describe the properties of their version chains, the experiment setup and the evaluation criteria. Our experiments are three-fold, to answer our research questions (Section 1).

4.1 Datasets

We use the following 139 RDF vocabulary version chains:

1 version chain of the DBpedia ontology [18] (8 versions), with community-curated classes and properties describing DBpedia

[2] Implementation and full results at http://goo.gl/rASX6S

Figure 1: Pipeline of our approach. Arrows show the data ow through the modules.

ID	Scope	Description
dirChildren	Per-version	Number of directly connected concepts via skos:broader, rdfs:subClassOf, etc.
dirChildrenD	Per-version	I.d. with direct children at 34?C <0G 4?C
parents	Per-version	Number of concepts this concept descends from
siblings	Per-version	Number of concepts that share parents with this concept
dirArticles	Per-version	Number of typed instances or user-de ned membership properties linking the concept with an instance (e.g. rdf:type)
dirArticlesChildrenD	Per-version	I.d. with children at 34?C <0G 4?C
ratioArticlesChildren	Per-version	Ratio of instances per number of direct children
ratioArticlesChildrenD	Per-version	I.d. with children at 34?C <0G 4?C
totalSize	Per-chain	Total size of the vocabulary chain in number of triples
nSnapshots	Per-chain	total number of vocabulary versions
avgGap	Per-chain	Average time gap (in days) between the versions
avgSize	Per-chain	Average size of the vocabulary across all versions in number of triples
nInserts	Per-chain	Average number of inserted new statements from one version to the next
nDeletes	Per-chain	number of deleted statements from one version to the next
nComm	Per-chain	number of common statements across versions
isTree	Per-chain	(True or false) Indicates whether the vocabulary is a tree (without cycles) or a graph (with cycles)
maxTreeDepth	Per-chain	maximum tree depth among versions (i.e., highest depth level of chained subconcept relations)
avgTreeDepth	Per-chain	Average of previous, across all versions
totalInstances	Per-chain	Total number of typed entities, i.e. instances belonging to a class through a membership property
ratioInstances	Per-chain	A0C8> =BC0=24C>C0; =BC0=42B>C0;(8I4
totalStructural	Per-chain	Total number of structural relationships, i.e. number of statements indicating the relationship between two vocabulary concepts
ratioStructural	Per-chain	A0C8>(CAD2CB>A0;C0;(CAD2CB>A0;C0;(8I4

Table 1: All features for vocabulary change considered in our approach, including per-version (i.e. one timestamped vocabulary snapshot) and per-chain (i.e. including all versions) features.

Figure 2: A Web vocabulary version chain. Training and test datasets for $\# = 7$, $= 1$ and $)) = 2$.

content. Instances are DBpedia resources with a rdf:type of some class in this ontology and some rdfs:label label.

1 version chain of the Dutch historical censuses dataset [28] (CEDAR, 8 versions), a SKOS taxonomy of historical occupations (HISCO). Instances are census observations with a dcterms:occupation of some HISCO concept and some skos:prefLabel label.

3 version chains from ontologies in the Linked Open Data cloud [33] (LODC, 3+ versions), returning 49,379 ontologies of which we lter all having at least two chained owl:priorVersion which are de-referenceable and parseable; this results in in 3 ontology chains (geonames, fao and lingvoj).

134 version chains from Linked Open Vocabularies [32] (LOV, 3+ versions), a well-known collection of Semantic WV (e.g. schema.org, PROV, DCAT, Bio, FOAF, etc.) that covers the 9 broad domains shown in Figure 3.

Each version within these chains consists of (a) schema information expressed using vocabularies like SKOS [25], RDF Schema [23] and OWL [19]; (b) instance data making use of such schema; and (c) labels describing the nodes of the schema and the instances. A detailed breakdown of these 4 groups, the 139 version chains and their characteristics is available online.[3] The selection of these speci c datasets is due to multiple reasons. First, they cover different levels of semantic expressivity, from SKOS taxonomies to OWL ontologies. Second, the temporal gap between each version varies from a minimum of 6 days (LOV), 10–12 months (DBpedia), 4.5 years (LOV) to 10 years (CEDAR). Third, the selection contains both manually and automatically created datasets: the DBpedia ontologies have a mixed automatic/manual maintenance [18], while the CEDAR data is a totally manually maintained dataset. Fourth, DBpedia data is born-digital, open, and still evolving (2007–), whilst

[3] http://bit.ly/kos-change

Figure 3: Domains represented in the LOV data. Dataset/metadata includes dataset, library, service descriptions. Government/organisations includes public and private descriptions. Cultural includes persons, social relations, media artifacts.

the CEDAR dataset is historical legacy, non born-digital, and temporally closed (1849–1930). The LOV dataset includes a brought range of sizes, on avarage 1K triples. LOV versions chains are on average considerably smaller than DBpedia and CEDAR, which have 10 million and over 2 million triples respectively. The three chains in LODC are of sizes 22K, 16.5K, and 1.5K triples each. LOV covers the range of small KOS starting at 19 triples for the smalles and ending at 18.6K triples for the largest. Conclusively, DBpedia, CEDAR, LOV and LODC cover a wide range of different KOS in various aspects such as size, ageing, and construction.

4.2 Experimental Setup

in accordance to our research questions our evaluation process is three-fold. First, we use selected features for supervised learning, and evaluate the quality of the resulting classi ers on predicting concept changes (RQ1). To evaluate the resulting models, we use 10-fold cross-validation on all datasets and report the performance. In addition, we use the long chains, and abundance of instance data, of DBpedia and CEDAR to compare the predictions with the actual changes in a next dataset version. To do this, we use the dataset produced after setting the parameter (t) , and we compare predictions with unseen labeled data. Since more versions are available in the chains of CEDAR and DBpedia, we execute several learning tasks adding more past versions from t_{-1} to t_{-8} incrementally. We study how this impacts prediction of change in t_1. To measure model performance we use precision, recall, F-measure, and area under the ROC curve. We report the best of the 10-fold cross-validation; in DBpedia and CEDAR, we also use the unseen t_{+1} version for testing. Second, we assess the quality of our features as concept change predictors, choosing the most performing ones via feature selection. Feature selection is executed by the ReliefF algorithm [15], whose results will answer RQ2. Third, we use linear and multimodal logistic regression on the learning results to explain what characteristics of the input versioned vocabularies make their changes more predictable. With this part of our evaluation, we answer RQ3.

Parameter tuning. For our framework parameters, we train models with all permutations regarding the number of ontology versions, the and paremters, and the values of structural, instance and label properties; and select the one with the best performance. For the ML hyperparameters, we rely on the default experimental values provided by the WEKA interface. The relevance threshold parameter is experimentally set at 0.8, the ReliefF parameters at t = 1, = 1, = 10, and the ranker parameters at = 0.05 and # = 1.

5 RESULTS AND DISCUSSION

We introduced a change prediction method based on concept change of previous versions. In this section, we report on our results, providing evidence for all our research questions evaluating: (RQ1) general performance using a concept change method to predict future changes in a domain-independent setting, aiming at genericity; and in order to understand and explain this genericity, (RQ2) the performance of the per-version feature set based on semantics (see Section 3.2) and structure of vocabularies; and (RQ3) the performance of the per-chain features leading to best change predictions and speci c model choices (see Section 3.2). We also discuss the results in connection with each research question separately. We nd: (RQ1) strong change prediction ML models for almost all datasets with >90% performance (Section 5.1); (RQ2) highly ranked per-version features dealing with the structure and instances of a concept and its neighbourhood (e.g. siblings, parents, dirArticlesChildrenD2) (Section 5.2); and (c) high correlations between per-chain explanatory variables such as Snapshots and avgGap, and performance of change prediction (Section 5.3).

We emphasise that change prediction, just like link prediction or graph completion methods, comes with its own challenges when evaluating for accuracy. True predictions are penalised for not being present in the dataset and assumed to be false. Therefore, the true performance cannot really be known [2, 26].

5.1 Change Prediction

Table 2 shows precision, recall, F-measure, and ROC scores after executing our proposed framework in all datasets described in Section 4.1. Predictions for DBpedia and CEDAR are reported for the newest version, for LODC and LOV we report CV results. The three chains in LODC (fao, geonames, lingvoj) are listed separately. As LOV is made of 134 different version chains, Table 2 shows the average over all of them. Predictive models for DBpedia and CEDAR achieve F-measures of 0.98 and 0.91 respectively. Figure 4 shows these results in more detail, running our approach six times on DBpedia and CEDAR to account for their long version chains. ROC areas show that models are robust using cross-validation in almost all datasets. Precision and recall are balanced and contribute equally to good F-measures. We observe how the non-over tting tendency of NaiveBayes is an advantage if the classi er is trained with more past versions: e.g. MultilayerPerceptron predicts better with less data (F-measures from 0.82 to 0.30), but with more versions NaiveBayes performs better (0.72 to 0.84). Overall, CV results across all 4 datasets, DBpedia (0.98), CEDAR (0.91), LODC (0.736) and LOV (0.922) are encouraging, and show that high performance (>90%) ML models for vocabulary change prediction can be learned using domain-agnostic features.

However, the availability of suf cient structural and instance data can severely affect these results, as shown in e.g. geonames (0.527). This might be especially true when using these models to predict the unseen t_{+1} versions, where performances can drop signi cantly (0.67

	DBpedia	CEDAR	fao (LODC)	geonames (LODC)	lingvoj (LODC)	LOV (avg.)
Precision	.98	.93	.751	.438	.95	.895
Recall	.98	.90	.765	.662	.947	.951
F-measure	.98	.91	.744	.527	.937	.922
ROC area	.81	.84	.844	.5	.792	.566
Best ML model	Random Forest	Simple Logistic	Hoeffding Tree	SGD	Multilayer Perceptron	Bayes Net

Table 2: Best 10-fold CV training scores in the version chains from DBpedia ($+_4$ = 2003), CEDAR ($+_4$ = 1930), the three LOD SPARQL endpoints (LODC), and average over LOV. Scores are between 0 (worst) and 1 (best). For LOV, BayesNet was chosen as the algorithm building the best models in the majority of chains.

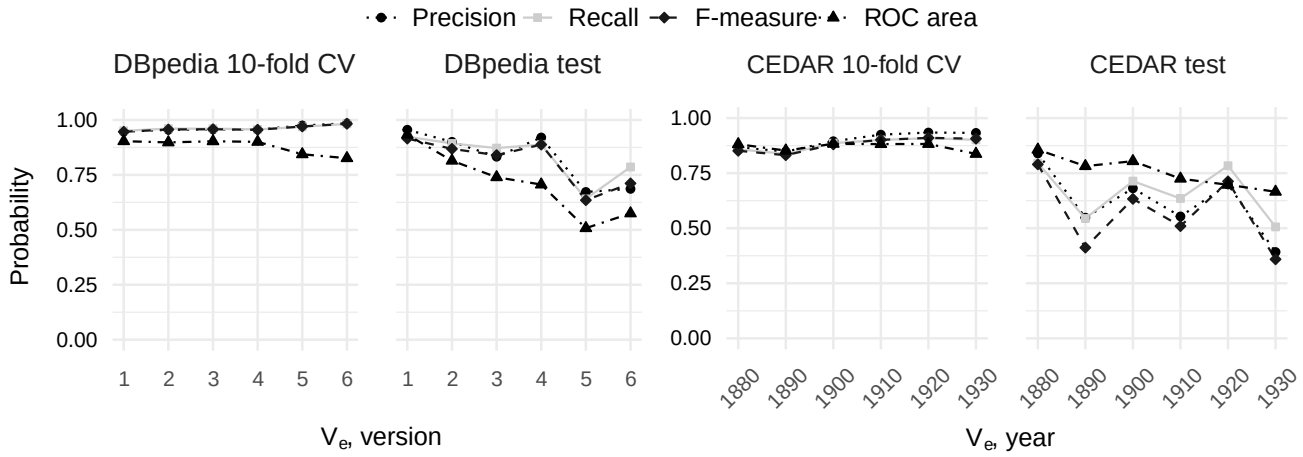

Figure 4: Best change prediction performance in the CEDAR and DBpedia refinement experiment with 6 incremental learning runs. The y-axis is the appropriate value for precision, recall, F-measure and ROC. The x-axis is a snapshot identifier.

in CEDAR and 0.36 in DBpedia). In CEDAR/DBpedia (Figure 4), we see a decreasing performance when the time gap increases; i.e. $+_4$ is harder to predict when it is farthest away in the future. A plausible explanation for this is that past versions contain valuable knowledge for change prediction only to an extent; after this, past knowledge would become noise if the vocabulary concepts have changed too abruptly in the last version $+_4$. We observe that CV results are generally worse in the LODC data (0.744, 0.527, 0.792). This can be due to the fact that datasets in SPARQL endpoints are generally less consistent, include less vocabulary descriptions, and are not always available [33]. Due to the unavailability of sufficient versions, LODC and LOV are only evaluated through 10-fold CV, with F-measures of 0.744, 0.527 and 0.937 for the LODC vocabularies (fao, geonames, lingvoj) and 0.922 for LOV. ROC areas show that models are robust using CV in all datasets, except for geonames (0.5).

Comparison with other methods. Some of these results are similar or outperform the state of the art [26]; simultaneously, they are very hard to compare to similar methods, especially SemaDrift [28] and FCA [11]. Comparing our results with [26], we observe that for DBpedia and CEDAR we clearly outperform in the 10-fold CV and the pure test results are also slightly higher or comparable. In the case of LOV biological ontologies, our 10-fold CV results match the performance of [26]. Additionally, we need to stress, that our task is not completely the same as [26]. We also found that our datasets are much smaller than GO. However, we do achieve similar and even better results with smaller datasets and shorter version chains. This indicates that our approach can be used without such an extensive evolution as seen with GO in [26], which is a positive finding. Additionally, our richer feature sets could also account for the better results. They make our approach more usable and explainable, because they are more general, do not restrict in the domain of application, and can be easily interpreted (e.g. see Table 3). At this point, we are unable to compare our results to those of SemaDrift [28] and FCA [11]. SemaDrift [28] is purely used for measuring concept drift between two concepts (not entire ontologies), but not for learning or predicting change; however we leave experimenting with SemaDrift's change metrics for future work. On the other hand, FCA [11] learns a new representation of the ontology and not a model of the changes which will occur. Another difference is the examined timeframes: [11] uses only a few weeks of data, where our dataset versions can be much further apart. They found that adding more data was beneficial to the performance of their model however, we found that forgetting too old versions increased our performance. Because of the different task, [11] old data is beneficial, whereas in our model older content can possibly include a shift in meaning making our performance worse.

CEDAR	DBpedia	LODC & LOV
siblings	dirChildren	parents
dirArticlesChildrenD2	siblings	siblings
ratioArticlesChildren	dirChildrenD2	ratio3
dirArticles	dirChildrenD3	dirChildren
dirArticlesD1	dirChildrenD4	ratio0
dirArticlesD2	dirArticlesChildrenD2	ratio

Table 3: Ranked features in CEDAR and DBpedia by `Relief` [15]. The ranking for LODC and LOV features is averaged from the rankings in specific vocabularies.

5.2 Per-version Features

Table 3 shows the ranking of selected per-version features according to their feature score in `Relief` [15]. Structural features are consistently ranked high across all datasets (e.g. `siblings`), while instance features are consistently important for the datasets that have them (e.g. `dirArticles`, `dirArticlesChildrenD2`).[4] In CEDAR, instance membership features (*dirArticles, dirArticlesChildren*) are more often selected. Conversely, in DBpedia structural properties are more often selected (*dirChildren, dirChildrenD, siblings*). In this same line, structural features are also preferential for LODC and LOV: *parents*, *siblings* and *dirChildren* rank higher, and seem to have more weight closer to the concept (*siblings*) than at higher depths (e.g. *dirChildrenD4*).

The availability of sufficient instance and schema data could explain the majority of the most influential per-version features. CEDAR, a taxonomy with instance data for almost every class, displays a preference for mixing both structural and instance features. DBpedia, is rich in both, but the complexity of the DBpedia ontology [30] might explain the preference for features about ontological relationships between classes and subclasses. The absence of instance data for LODC and LOV makes the choice for structural features preferential as the main signal for change prediction. The unavailability of instances in LODC and LOV seem to indicate the importance of structural features.An explanation could be that vocabularies in LODC and LOV are generally more flat and hence changes generally have an impact on specific, sparse concepts.

A qualitative evaluation on specific vocabulary changes supports these hypotheses. In CEDAR, `cedar:hisco-06`, the class of "medical, dental, veterinary and related workers", is a concept correctly predicted to change. Most of its structural features present high stability across versions, e.g. number of children (4) and siblings (9); but those related to its instances vary, e.g. number of instances (841, 68, 143, 662, 110). In DBpedia, the concept `dbpedia:CollegeCoach` is also expected to change, with the number of Wikipedia articles pointing to it (instance feature) increases linearly (2787, 3520, 4036, etc.). Structurally, however, its siblings remain stable (21, 21, 23, 23) until it gets a new parent and its siblings suddenly explode (23, 344). This shows that both types of features can be deciding in terms of change, and the importance of feature engineering for generating human-understandable explanations.

Therefore, instance features are ranked higher to predict change in datasets that have rich information on them (CEDAR); while structural features are preferential in datasets with more structural (LODC, LOV) and hybrid (DBpedia) information. In other words, selected features and classifiers depend on the kind of data available; however, when both are present (e.g. DBpedia), structural features continue to be more defining of change than instance ones.

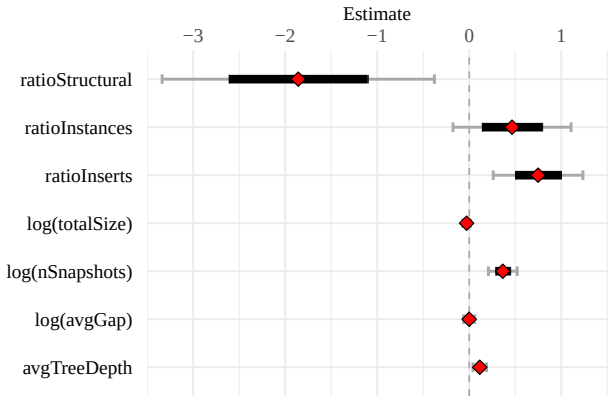

Figure 5: Coefficient values of the best linear regression model explaining change predictability depending on features.

5.3 Per-chain Features

To explain per-chain features (second half of Table 1) that impact change predictability we use: (a) regression analysis, to understand which ones are good predictors of high classification performance (i.e. we use area under the ROC curve as a response variable); and (b) multinomial logistic regression, to analyse which ones are good predictors of the best classifier type. The best regression analysis model is shown in Figure 5. Since we have 137 degrees of freedom, at $? < 0"05$ significance level any A value above 0.166 denotes dependency. *nSnapshots* (0.276) and *avgTreeDepth* (0.192) hold a direct dependency on *roc*, while *avgGap* (-0.263) holds an inverse dependency. Larger versions (*totalSize*, 0.180) with more inserts between versions (*nInserts*, 0.166) and more instances (*totalInstances*, 0.176) explain models with high precision and recall. In summary, version chains with more versions (snapshots), more frequent releases, with deeper tree structures and with more instance data are related to better predictive models for vocabulary change. We find that *avgGap* is influential at selecting a tree-based classifier instead of a bayes-based one. According to the results of the multinomial logistic regression,[5] we find that *totalSize* is influential at selecting function- and rule-based classifiers instead of bayes-based classifiers. Almost all classifier families will be less likely chosen if the elapsed time between versions (*avgGap*) increases; in other words, more frequent releases will favour most models predicting vocabulary change. Interestingly, ratios on instance and schema data will influence the best classifier type in an inverse way: more instance data will favour tree-based and rule-based classifiers; while more schema data will favour bayes-based classifiers. Finally, multinomial logistic regression shows that the most performant classifier is selected mostly depending on the number of versions in the chain, the tree depth of these versions, and the ratio of instance data in each version.

[4] We leave a more formal analysis using feature consistency metrics for future work.

[5] See supplementary material at http://bit.ly/web-vocabulary-change