Predicting Concept Drift in Linked Data

Albert Meroño-Peñuela[1,2] and Christophe Guéret[2]

[1] Department of Computer Science, VU University Amsterdam, NL
albert.merono@vu.nl
[2] Data Archiving and Networked Services, KNAW, NL

Abstract. The development and maintenance of Knowledge Organization Systems (KOS) such as classification schemas, taxonomies and ontologies is a knowledge intensive task that requires a lot of manual effort. Librarians, historians and social scientists, among others, constantly need to deal with the change of meaning of concepts (i.e. *concept drift*) that comes with new data releases and novel data links. In this paper we introduce a method to automatically detect which parts of a KOS are likely to experience concept drift. We use supervised learning on features extracted from past versions to predict the concepts that will experience drift in a future version. We show that an existing domain-specific approach for predicting the extension of concepts can be successfully generalized to predict concept drift in KOS in a domain-independent manner. This result is confirmed by our experiments on a variety of datasets (encyclopedic, cross-domain, and socio historical), enabling the creation of a generic tool to assist knowledge experts in their curation tasks.

Keywords: Concept Drift, Ontology Change, Ontology Evolution

1 Introduction

Motivation. Knowledge Organization Systems (KOS), such as SKOS taxonomies and OWL ontologies, play a crucial role in the Semantic Web. They are at the core of any Linked Data vocabulary and provide structured access to data, formalize the semantics of multiple domains, and extend interoperability across the Web. *Concepts* are central entities in KOS and represent objects with common characteristics. However, with time, objects are continuously subject to change. As the world changes, our understanding of it evolves. Concepts in KOS are inherently affected by *concept drift*, the change of their meaning over time [18].

What is the problem? Curation is a manual, knowledge intensive and arduous task. As domain concepts drift their meaning, curators and maintainers release different dataset *versions* updating their KOS, mostly using their expert knowledge. Concept drift positions them in a great dilemma: the continuous evolution of knowledge has a severe impact on their update work. Automatic detection of concept drift in their curation process would be a great aid. Unfortunately, there is currently hardly any tool to help dealing with concept drift.

Use-cases. To enhance comparability studies in the history of work, social scientists and historians have developed the Historical International Standard

Classification of Occupations (HISCO), a taxonomy of tens of thousands of occupations from countries and languages around the world since the 16th century[3]. Many historical datasets update their mappings to HISCO across their different versions. The Gene Ontology (GO) standardizes the representation of gene attributes across species and datasets. A new version is released every month.[4] Wikipedia categories and the DBpedia ontology are updated to improve cross-domain knowledge access. Librarians, social scientists, historians or biologists have a dire need to assess this versioning processes by identifying, and predicting, when a concept, class or code will change in the forthcoming release. This will reduce time spent in manual data exploration or requirements gathering.

Contribution. This paper introduces a generic approach to predict concept drift. Previous approaches have proven to be effective in predicting ontology *extension* in the *biomedical domain* [15]. We take this work as a basis to develop a *domain independent concept drift detection framework* based on supervised learning on past versions of Linked Datasets. We build a generic tool to aid data curators in maintaining datasets and increasing their proactivity towards change over time. This is useful in two scenarios: *refinement*, i.e. helping curators to revise a past version to check its coherence; and *prediction*, i.e. helping curators to be aware of concept drifts to come, allowing them to give higher maintenance priority to concepts that are more likely to change than others. We show viability of our approach in social sciences, history, and encyclopedic and cross-domain knowledge, adding to the already investigated biosciences [15].

Research Questions. In order to show validity of our approach we focus our research on the following three concrete research questions:

- **RQ1**. Can past knowledge be used to predict concept drift? Can this be done by extending an extension prediction method ([15]) into a concept drift prediction method?
- **RQ2**. What features encoding past knowledge have a greater influence on future drifts? What classifier performs best to predict these drifts?
- **RQ3**. Can this new method predict concept drift in a domain-independent manner? Can our algorithms predict drift on any Linked Data dataset?

Findings. In order to answer the research questions we run two refinement experiments in the Dutch historical censuses and the DBpedia ontology datasets, and a prediction experiment in the DBpedia ontology dataset. We obtain solid refinement/prediction performances, with f-measures of 0.84, 0.93 and 0.79 on test data in these experiments. We foresee a useful and effective domain-independent concept drift detection tool to aid data curators in their versioning processes.

The rest of the paper is structured as follows. In Sections 2 and 3 we survey previous efforts to address change, and define our target problem and formalism. Section 4 describes our approach, pipeline and feature set. In Section 5 we perform an experimental evaluation, describing the input data, process and results. In 6 we discuss these results and their relation with our research questions, before we conclude in Section 7.

[3]See `http://historyofwork.iisg.nl/`

[4]See `http://www.geneontology.org/`

2 Related Work

Concept drift is a very active research topic in Machine Learning. It is difficult to learn in real-world domains 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*” [17]. Hence, drift occurs in a concept when the statistical properties of a target variable (the *concept*) change over time in unforeseen ways. Multiple concept drift detection methods have been developed [5].

With the advent of the Semantic Web, changes in concepts have been investigated by formally studying the differences between ontologies in Description Logics [6]. Fanizzi et al. [3] propose a method based on clustering similar instances to detect drifts. Wang et al. [18] define what concept drift is and how to identify it in a Semantic Web setting. The related field of *ontology evolution* deals with “the timely adaptation of an ontology and consistent propagation of changes to dependent artifacts” [1]. As stated by Stojanovic [16], the first step for any evolution process consists in identifying the need for change; change capturing can then be studied as *struture-driven*, *data-driven* or *usage-driven*. 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” [4,10,7]) suggests that the overlap between the two fields is considerable. Pesquita and Couto [15] develop the closest match to our work. They propose a method based on supervised learning on past ontology versions to predict extension of biomedical ontologies, using Stojanovic’s guidelines to design good predictors of change.

The need of tracing concept drift in application areas of the Semantic Web has been stressed, particularly in the Digital Humanities [13] and Linked Statistical Data, where concept comparability [2] and extensional drift [14] are key.

3 Problem Definition

3.1 Concept Drift

We use the framework of concept drift proposed by Wang et al. [18] in order to provide a definition and formalism.

Definition 1. *The* meaning of a concept *C is a triple (label(C),int(C),ext(C)), where label(C) is a string, int(C) a set of properties (the* intension *of C), and ext(C) a subset of the universe (the* extension *of C).*

For the specific goal of this paper, we define the intension of a concept $int(C)$ as the set of structural relationships of C with other concepts, indicated by a set of user-defined predicates (e.g. `skos:broader`, `rdfs:subClassOf`). We define the extension of a concept $ext(C)$ as the set-members of C. We define the label of a concept $label(C)$ as the set of labels given to C.

All the elements of the meaning of a concept can change. To address concept identity over time, authors in [18] assume that the intension of a concept C is the disjoint union of a rigid and a non-rigid set of properties (i.e. $(int_r(C) \cup int_{nr}(C)))$. Then, a concept is uniquely identified by some essential properties that do not change. The notion of identity allows the comparison of two variants of a concept at different points in time, even if a change on its meaning occurs. For the specific goal of this paper, we assume that concepts with identical rigid intensions will have the same URIs[5].

Definition 2. *Two concepts C_1 and C_2 are considered identical if and only if, their rigid intensions are equivalent,* i.e., *$int_r(C_1) = int_r(C_2)$.*

If two variants of a concept at two different times have the same meaning, there is no concept drift. Intensional, extensional, and label similarity functions $sim_{int}, sim_{ext}, sim_{label}$ need to be defined to quantify meaning similarity. These functions have range $[0, 1]$, and a similarity value of 1 indicates equality. For the specific goal of this paper, we define $sim_{int}(C', C'')$ as the function that returns whether C' and C'' have the same direct structural features. We define $sim_{ext}(C', C'')$ as the function that returns whether C' and C'' have exactly the same cardinality or not. We define $sim_{label}(C', C'')$ as the function that returns the minimum edit distance between the set of all labels assigned to C' and the set of all labels assigned to C''.

We implement this framework as our definition of concept drift, and we instantiate it by concretizing the mentioned sets and functions for the specific goal of this paper. Our approach aims at genericity and these choices, as well as the overall drift definition, are cusotmizable by the user (see Section 4).

Definition 3. *A concept has extensionally drifted in two of its variants C' and C", if and only if, $sim_{ext}(C', C'') \neq 1$. Intensional and label drift are defined similarly.*

3.2 Usage

We study concept drift in two different scenarios, depending on whether the versioned Linked Dataset is *closed* or *open*. We define a *closed dataset* as the dataset which versions were released in the past and their production finished at a certain point; consequently, no new versions will come. We define an *open dataset* as the dataset which updated versions are currently under production; new versions are expected to come.

[5]We are aware that, in the general context of the Semantic Web, this assumption is not true: different versions of concepts should get distinct URIs, while different URIs may point to exactly the same concept due to the distributed nature of the Web. In general, users need to define their own identity functions between concept versions, e.g. using `owl:sameAs` or `skos:exactMatch` mappings.

Refinement. Given that no future versions will come, the prediction of concept drift in closed datasets is meaningless. In this scenario, we propose *refinement*. The purpose of refinement is to check the coherence of an intermediate version (instead of a version to come) with respect to previous versions.

Prediction. In open datasets, the primary question is to predict concept drift in a next version. To address this we propose the *prediction* usage. In prediction we use all available past versions to train a classifier and predict which concepts will drift in a next release. Obviously, *refinement* can also be applied to open datasets to check their internal coherency.

4 Approach

The basic assumption of our proposed approach is that the knowledge encoded in past versions of a Linked Dataset can be used to faithfully predict which parts of it will suffer concept drift in a forthcoming version. This idea is inspired by the work by Pesquita and Couto [15] on predicting the extension of biomedical ontologies. Their approach derives from change capturing strategies that are based on implicit requirements. Traditionally, these requirements are defined manually on an expert knowledge basis. Their work tries to learn these requirements from previous extension events by selecting features that, according to Stojanovic [16], may have an influence in changing parts of an ontology:

– **Structure-driven**, derived from the structure of the ontology (e.g. if a class has a single sublcass, both should be merged).
– **Data-driven**, derived from the instances that belong to the ontology (e.g. if a class has many instances, the class should be splitted).
– **Usage-driven**, derived from the usage patterns of the ontology in the system it feeds (e.g. remove a class that has not been accessed in a long time).

The state of the art [15] has proven success in the use of these features (i) to predict *extension*, that is, to estimate if a class will be extended (e.g. with new children or properties) in the future; (ii) in (OBO/OWL) ontologies; and (iii) in the biomedical domain (using the Gene Ontology). However, it remains unclear if supervised learning can be generally applied (I) to predict *drift*, that is, to estimate if a concept will experience change in its meaning; (II) in any Linked Dataset (i.e. generic RDF graphs); and (III) in a domain-independent manner.

Consequently, we extend the three essential aspects (i), (ii), (iii) of [15] in order to clarify the more generic (I), (II), (III) issues. These extensions, as we show in this Section, are not trivial. In the following we present a pipeline that includes: (a) an abstraction of the input parameters required for the learning process; (b) an abstraction of features that apply not only to OBO/OWL ontologies, but to any Linked Dataset; and (c) a pre-learning optimisation technique to merge features of identical versioned concepts into single training/test individuals. As a result, our approach tackles **RQ1** by providing generic and customizable change definition functions (including those in [18] (see Section 3.1)); generic and customizable features, including free choice of predicates/RDF types

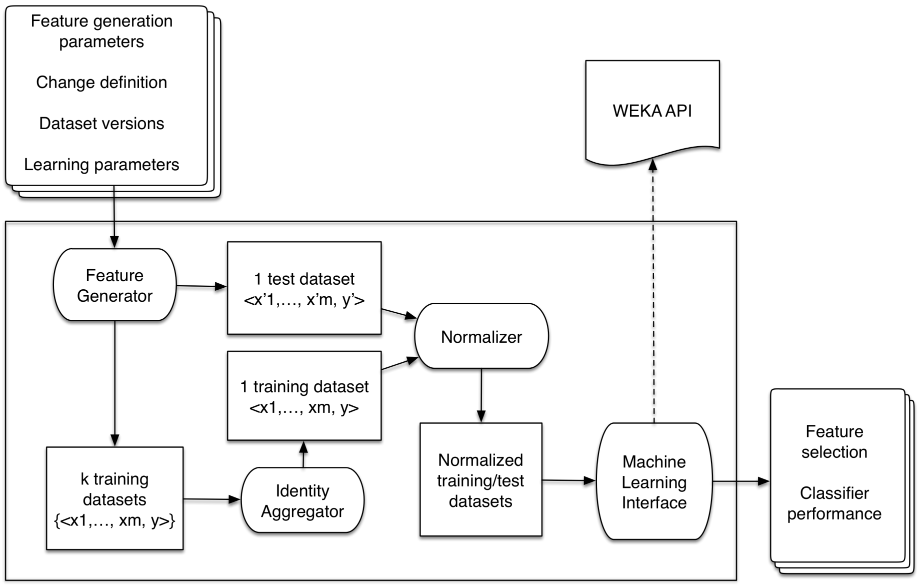

Feature generation parameters
Change definition
Dataset versions
Learning parameters
WEKA API
Feature Generator
1 test dataset <x'1,..., x'm, y'>
Normalizer
1 training dataset <x1,..., xm, y>
k training datasets {<x1,..., xm, y>}
Identity Aggregator
Normalized training/test datasets
Machine Learning Interface
Feature selection
Classifier performance

Fig. 1: Pipeline of our approach. Arrows show the data flow through the modules.

to use in their generation; customizable machine learning algorithms (feature selection and classification); and systematic and fully automated executions –from input Linked Dataset versions to output feature/classifier performances.

4.1 Pipeline

Figure 1 shows the pipeline of our proposed approach. The data flow starts in the upper left corner: the system gets the input set {*Feature generation parameters, change definition, dataset versions, learning parameters*}, and returns the output set {*Feature selection, classifier performance*} in the bottom right corner.

The first module is the **Feature Generator (FG)**. It generates k training datasets and one test dataset, according to the following input set elements: (a) (N) *dataset versions*, in any RDF serialization, where the concept drift prediction is to be performed; (b) several user-set *feature generation parameters* that control the feature generation process (the $\boldsymbol{\Delta}$**FC** parameter, setting the version to be used to decide if a concept of the training dataset has drifted or not; and the $\boldsymbol{\Delta}$**TT** parameter, setting the version to be used to decide if a concept of the test dataset has drifted or not); and (c) a customizable *definition of change* that determines the value of the target variable. The last element of the input set, *learning parameters*, is passed further to be used in a later stage.

Once all set, k training datasets and the test dataset are built by the FG as shown in Figure 2. The parameters N, ΔFC and ΔTT are used to determine which versions will play the role of $\{V_t\}$, V_r and V_e. $\{V_t\}$ is the set of *training versions*, which are used to build the training dataset. V_r is the *reference version*, against which all versions in $\{V_t\}$ are compared, using the *definition of change*

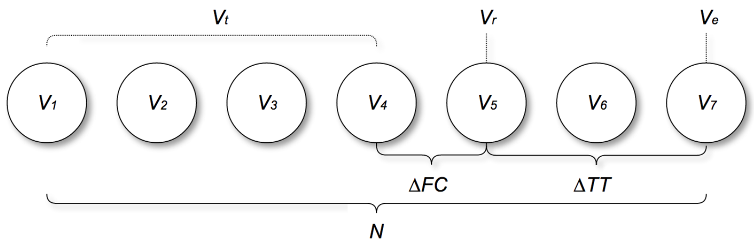


Fig. 2: Building the training and testing datasets for $N = 7$, $\Delta FC = 1$ and $\Delta TT = 2$.

provided as input, to determine whether there is concept drift or not. V_e is the *evaluation version* and is used to build the test dataset, following a similar procedure as with $\{V_t\}$ and V_r, this time comparing V_r with V_e. V_e is set by default to the most recent version. While extracting the features for each concept in the N versions, each exemplar is labeled depending on whether concept drift happened between one version of the concept and the next. This set of labels defines the target variable for the classification task. Our framework aims at genericity, and users can easily define their own notion of change and drift. We use the concept drift definitions described in Section 3.1 to decide if a concept drifts or not between two versions.

The FG produces k training datasets because versions can only be compared pair by pair. Although appending these k datasets into a single one is a possibility, it is important to preserve identity of the learning instances. Solving this is the purpose of the **Identity Aggregator (IA)**. The IA matches individual concepts of the k training datasets and merges their features into a single individual, modifying the dataset dimensionality accordingly. To do this, we match the URIs of the individual concepts; if matching concepts cannot be found, we fill features with `NA` values and modify the target variable value accordingly.

The single training and test datasets are then ingested by the **Normalizer (Norm)**. Norm adjusts value ranges, recodes feature names and types, and discards outliers, producing normalized versions of both datsets.

Finally, the training and test datasets are used by the **Machine Learning Interface (MLI)** as an input for the feature selection and classification tasks. These are done in a generic and customizable way, building on top of the implementation of state-of-the-art machine learning algorithms contained in the WEKA API [8]. The last element of the pipeline's input set, *learning parameters*, is used here to achieve this and contains: (a) the specific feature selection algorithm (from those included in WEKA) that ranks the features according to their relevance to predict drift; (b) a threshold t; only features with a relevance higher than t will be selected; and (c) the list of classifiers to be trained. First, the module runs the chosen feature selection algorithm to select the most relevant features. Secondly, it trains systematically any subset of all WEKA classifiers included in the WEKA API, as chosen (by default all classifiers are trained). Lastly, it evaluates the trained models, and stores the results in disk.

4.2 Feature Set

We propose a set of features based on *structural properties* and *membership properties* of concepts as attributes for learning.

Structural features measure the location and the surrounding context of a concept in the dataset schema, such as children concepts, sibling concepts, height of a concept (i.e. distance to the leaves), etc. Since classification schemas are graphs in general and may contain cycles, these properties are defined with a *maxDepth* threshold that indicates the maximum level at which the property will be calculated (e.g. direct children, children at depth one, two, etc.). A concept is considered to be a child of another if they are connected by a user-specified property (e.g. `skos:broader`, `skos:narrower` or `rdfs:subClassOf`). We use *direct children* (descendants at distance 1) [*dirChildren*], *children at depth* $\leq$ *maxDepth* [*dirChildrenD*], *direct parents* (concepts this concept descends from) [*parents*], and *siblings* (concepts that share parents with this concept).

Membership features measure to what extent a concept in the classification is used in the data. A data item in a Linked Dataset is considered to be using a concept of the classification if there is a user-defined membership property linking the data item with the concept (e.g. `dc:subject` or `rdf:type`). We use *members of this concept* [*dirArticles*] and *total members considering all children at depth* $\leq$ *maxDepth* [*dirArticlesChildrenD*] as membership features. Finally, we define a set of hybrid features that combine the previous features into a single value (e.g. ratio of members per number of direct children) [*ratioArticlesChildren*, *ratioArticlesChildrenD*]. These sets of features map conveniently to the different types of change discovery described by Stojanovic [16]: structural features implement structure-driven change discovery; and membership features can be seen both as data-driven (since they describe instances belonging to the ontology) and usage-driven (since users querying these are indirectly using their classes).

These features are computed for each concept in all versions as indicated by the training and test dataset building parameters (see FG module, Section 4.1). However, not all of them may be used for predicting concept drift. [15] shows that similar features based on Stojanovic's criteria [16] are good candidates. However, since it is unclear whether or not these features explains well enough concept drift in arbitrary domains, we only use those that prove to be good predictors of concept drift in the feature selection phase (see MLI module, Section 4.1).

5 Evaluation

We apply our proposed approach to different versioned Linked Datasets[6]. We describe the properties of such datasets, the experiment setup and the evaluation criteria. We report on our results, providing evidence to **RQ2**: we evaluate (a) the performance of the feature set as a generic predictor of concept drift (see Section 4.2) and (b) the performance of the classifiers at the predicting task.

[6]See implementation details and extended results at `http://goo.gl/rASX6S`

5.1 Input Data

In order to study the genericity of our approach and its applicability in a domain-independent setting, we use a set of multi- and interdisciplinary Linked Data datasets for which several versions have been published. Concretely, we use the 8 latest versions of the DBpedia ontology , and the last 8 versions of the Dutch historical censuses dataset (CEDAR)[7]. Each version consists of (a) a schema in SKOS, RDFS or OWL; (b) a set of data items making use of such schema; and (c) a set of labels describing the nodes of the schema.

In the case of the DBpedia ontology, the classification is the DBpedia OWL ontology itself [12], a community-curated formalization of all classes and properties describing DBpedia content. The data items are all resources of DBpedia which have some class of the ontology as `rdf:type`. The set of labels are the `rdfs:label` literals attached to the classes of each versioned ontology. In the case of the Dutch historical censuses (CEDAR) dataset, the classification is a SKOS hierarchy of HISCO occupations reported in each version. The data items are all census observations of people having one of these HISCO occupations as `cedar:occupation` (this is, the number of people employed in a certain occupation in a given version). The set of labels are the `skos:prefLabel` (Dutch) literals used in the census to describe these occupations in each specific version. A summary of the selected datasets is shown in Table 1.

Dataset	Version	N. concepts	N. relations	N. members
DBpedia ontology	(3.3) May 20th, 2009	282	174	3.8M
	(3.4) September 24th, 2009	334	204	4.3M
	(3.5) March 16th, 2010	255	255	5.2M
	(3.5.1) March 16th, 2010	257	257	5.5M
	(3.6) October 11th, 2010	272	272	6.2M
	(3.7) July 22nd, 2011	319	610	9.3M
	(3.8) June 1st, 2012	359	369	13.2M
	(3.9) April 3rd, 2013	529	541	15.9M
CEDAR	November 19th, 1849	408	405	76K
	December 30th, 1859	310	307	16K
	December 1st, 1869	238	236	7K
	December 31st, 1889	546	543	723K
	December 31st, 1899	566	563	837K
	December 31st, 1909	685	682	334K
	December 31st, 1920	271	268	103K
	December 31st, 1930	584	581	78K

Table 1: Summary of the selected datasets. *Version* refers to the date in which a specific version was published; *N. concepts* is a count of concepts/classes in that version; *N. relations* is a count of relationships connecting these concepts/classes; and *N. members* is the number of individuals making use of each classification.

The selection of these specific 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 10-12 months (in DBpedia) to 10 years (in CEDAR). Third, the selection contains

[7]See `http://www.cedar-project.nl/`

both manually and automatically created datasets: the DBpedia ontologies have a mixed automatic/manual maintenance [12], while the CEDAR data is a totally manually maintained dataset. Fourth, DBpedia data is born-digital, open, and still evolving (2001–), whilst the CEDAR dataset is historical legacy, non born-digital, and temporally closed (1795–1971).

5.2 Experimental Setup

Due to their inherent nature, CEDAR is a *closed* dataset, while DBpedia categories and DBpedia ontology are *open* datasets (see Section 3.2). For this reason, we divide our experimental setup in two parts: two *refinement* concept drift experiments using the CEDAR data (closed) and the DBpedia ontology (open); and one *prediction* concept drift experiment using the DBpedia ontology (open).

We set all parameters (see Section 4.1) as demanded by this setup. Since the datasets present temporal uniformity (see Table 1), we set $\Delta FC = \Delta TT = 1$.

Our evaluation process is two-fold. First, we quantify the quality of our features as concept drift predictors; we then choose the best performing features. We do this via *feature selection* (see Section 4.1). Second, we use this performant subset to evaluate the average quality of classifiers on predicting concept drift; we then choose the best performing classifiers.

To evaluate classifiers we follow a simple approach: we compare the predictions made by the classifiers with the actual concept drift going on in a next dataset version. To do this, we use the test dataset (see Section 4.1) produced after setting the parameter ΔTT. Since we compare predictions with unseen labeled data, we know whether the predictions are correct or not.

We evaluate the refinement and prediction experiments differently. Since the goal of refinement is to check coherence of the drift of concepts in an intermediate version, we execute several learning tasks adding more past versions to $\{V_t\}$ incrementally. We study how this impacts prediction of drift in V_i. The prediction experiments, on the other hand, consider all available versions, and we use the trained classifiers to predict concept drift in the most current version.

For assessing model quality, we use standard performance measures in machine learning: precision, recall, f-measure, and area under the ROC curve . Using these measures, we perform a two-fold evaluation. On one hand, we evaluate the quality of the models produced without making any predictions and using 10-fold cross-validation with the training data. In 10-fold cross-validation, the training set is divided in 10 equal folds: 9 are used to train the classifier, and the remaining one to test predictions (this is averaged on 10 repetitions in order to use all folds as test folds). On the other hand, we use the same indicators to evaluate the performance of prediction of the trained classifiers using the unseen test datasets V_e/V_i. In order to get insight on the prediction performance, we compare these measurements with a baseline implementing random prediction.

Feature	Selection freq.
siblings	10
dirArticlesChildrenD2	8
ratioArticlesChildren	8
dirArticles	8
dirArticlesD1	8
dirArticlesD2	6

(a) Top selected features for CEDAR.

Feature	Selection freq.
dirChildren	21
siblings	21
dirChildrenD2	20
dirChildrenD3	20
dirChildrenD4 4	20
dirArticlesChildrenD2	17

(b) Top selected features for DBpedia.

Table 2: Top selected features in the CEDAR and DBpedia datasets.

5.3 Results

Selected features for CEDAR and DBpedia Table 2 shows the top selected features by the `Relief` algorithm [9], included in the WEKA API (see Section 4.1). The features are ordered according to the frequency in which they are selected across the versions. Table 2a shows that membership features (*dirArticles, dirArticlesChildren*) are systematically selected in the CEDAR data instead of structural properties (*siblings, dirChildren*). On the other hand, table 2b shows a clear preference for structural properties (*dirChildren, dirChildrenD, siblings*) in the DBpedia data, although some membership features have also an important relevance.

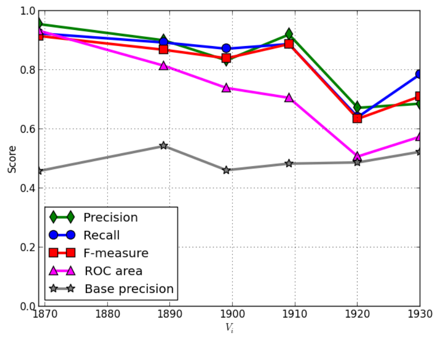
Refinement on CEDAR and DBpedia ontology We perform a refinement (see Section 3.2) experiment using the CEDAR dataset. We execute our approach six times, adding one Linked Dataset version to $\{V_t\}$ and shifting V_i forward once each time. We identify each experiment with the year of the version to be refined $V_i \in [1869, 1889, 1899, 1909, 1920, 1930]$. Figure 3 compares the classification results for all runs.

We perform a refinement experiment with the DBpedia ontology versions. We execute our strategy six times over the eight versions, first taking the first three, and adding one to $\{V_t\}$ and shifting V_i once each time (see Section 3.2). Figure 4 shows the results, which can be compared with the results of the previous experiment shown on CEDAR data in Figure 3.

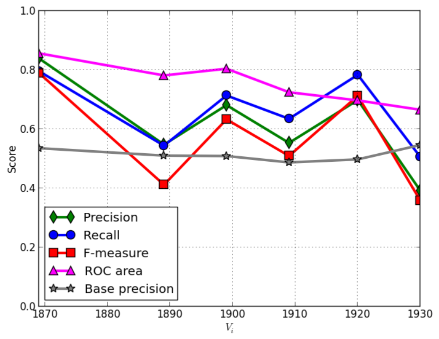
Prediction on DBpedia ontology We perform a prediction (see Section 3.2) experiment on the DBpedia ontology. We execute our approach once, using all available versions as training set $\{V_t\}$, and leaving the last for testing (V_e). Table 3 shows the performance of classification.

6 Discussion and Lessons Learnt

Table 2 shows that selection of features of one kind or another is influenced by the nature of the dataset under consideration. On the one hand, membership features are ranked as the most important ones to predict concept drift in CEDAR (i.e. knowledge on how instances are linked to the schema). On the other hand, structural features are top ranked to predict drift in DBpedia (i.e. knowledge

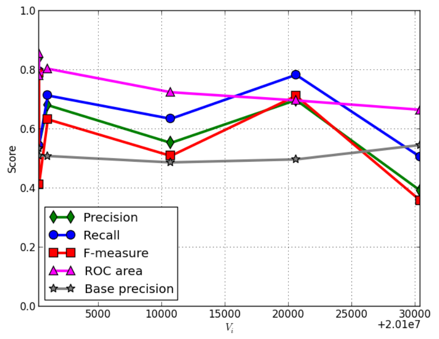


(a) 10-fold CV scores on the CEDAR training dataset.

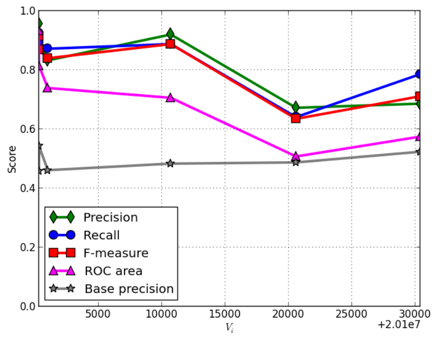


(b) Prediction scores on the CEDAR test dataset.

Fig. 3: Average classifier performance in the CEDAR refinement experiment with 6 incremental learning runs. Lines show performance measures varying along them.



(a) 10-fold CV scores on the DBpedia ontology training dataset.



(b) Prediction scores on the DBpedia ontology test dataset.

Fig. 4: Average classifier performance in the DBpedia ontology refinement experiment with 6 incremental learning runs. Lines show performance measures varying along them.

about the basic structure of the ontology). Both kinds of features are selected for both datasets at some point, though.

Figures 3 and 4 show that classification outperforms the simple random baseline. Although the Logistic, the MultilayerPerceptron and the tree-based algorithms have good performance in specific situations, the NaiveBayes classifier shows consistent results in all concept drift refinement and prediction experiments. Similar behavior and results have been described in [15]. Interestingly, we observe how the non-overfitting tendency of NaiveBayes is an advantage if the classifier is trained with more past versions. MultilayerPerceptron, for instance, predicts better with less data (f-measures from 0.82 to 0.30), but with more versions NaiveBayes wins (0.72 to 0.84) (see Figure 3).

In the refinement experiments performance is affected by two factors: the amount of past versions used to learn, and the characteristics of these versions versus the version to predict. In general we observe more performant models at

$V_e =$	**2013**
Precision	.98
Recall	.98
F-measure	.98
ROC area	.81
Base precision	.48

(a) DBpedia ontology 10-fold CV scores, training dataset.

$V_e =$	**2013**
Precision	.66
Recall	.75
F-measure	.67
ROC area	.58
Base precision	.52

(b) DBpedia ontology prediction scores, test dataset.

Table 3: Average classifier performance of prediction in the DBpedia ontology.

predicting V_i when more versions are added to $\{V_t\}$. However, CEDAR's 1889 and 1930 versions DBpedia ontology's 3.5.1 and 3.7 are clear exceptions (see Figures 3b and 4b). This can be due to several reasons. First, the version V_i to be predicted may contain unexpected changes that have not been learned from previous versions. CEDAR versions of 1889 and 1930, for instance, are known to had suffered a major restructuring or revision almost from scratch [11], making their drifts harder to predict. Second, corner cases of concept drift might not be captured with the feature set. Third, these CEDAR versions contain scarce member data (see Table 1), and lack of data about uncommon types of drift seems to hold a relationship on why prediction is harder in these cases. Still, our refinement approach proves to be useful on detecting these coherence data-issues.

Results also confirm that forthcoming concept drifts depend on past drifts, and that incorporating past knowledge about drift helps on predicting it. However, abusing this has a side effect, since adding too many old versions to the learning set $\{V_t\}$ seems to lower the prediction performance. This suggests that the inclusion of a moving window over $\{V_t\}$ could be a sensible solution.

`http://cedar.example.org/ns#hisco-06` is an example of a CEDAR/HISCO concept predicted to drift which in fact did: the class of "medical, dental, veterinary and related workers". Most of its features present high stability across the versions; except those related to its members, i.e. the sets of people reported to belong to this concept. These vary from 841 sets of observations, to 68, 143, 662 and 110, while structural properties like number of children (4) or siblings (9) remain relatively stable. `http://dbpedia.org/ontology/CollegeCoach` is an interesting DBpedia concept also expected to drift. The number of Wikipedia articles pointing to it increases linearly (2787, 3520, 4036, 4870...); it always remains a leave with a unique parent, so its children subhierarchy does not change either. Interestingly, its siblings remain stable (21, 21, 23, 23) until it gets a new parent and its siblings suddenly explode (23, 344). Considering these observations, it is easy to see why membership and structural features are highly ranked for CEDAR and DBpedia, respectively.

According to this, an important lesson to be learnt is that *the degree of dependency between the changed parts of a dataset and well-known change-aware features can save enormous amounts of update work.* We have shown that these features are collectable, rankable and usable in a fully-automatic and customizable pipeline to predict future drifts in Linked Data. If this degree of dependency is high, knowledge engineering tasks can be automated (by the classifier) to avoid coherency mistakes or taking too narrow decisions.

7 Conclusions and Future Work

Concept drift poses enormous challenges to data curators and maintainers. For those publishing their data as Linked Data, change of meaning of concepts implies the continuous release of versions, which is a knowledge-based and labor-intensive task. Our approach detects automatically which parts of a Linked Dataset will likely undergo concept drift in a forthcoming version, using supervised learning and leveraging drift knowledge contained in past versions.

Recalling back our research questions and results, the assumption that concept drift can be predicted in refinement/prediction scenarios using past knowledge is acceptable considering intensional, extensional and label drifts. We have shown how to achieve this by generalizing the state of the art in a Machine Learning for Linked Data pipeline (**RQ1**, see Section 4). We have studied the variance in relevant features from our feature set, and how classifiers behave using these features to predict concept drift (**RQ2**, see Section 5). With respect to its domain-independent applicability (**RQ3**), we predict drift accurately using non-domain related features in sociohistorical and cross-domain datasets, which add to the list of biomedical ontologies [15]. We get encouraging prediction results, obtaining an f-measure of 0.84, 0.93 0.79 for our selected classifier when predicting unseen data using sociohistorical, encyclopedic and cross-domain datasets. We study the impact of the addition of more past knowledge, and the limitations of the approach regarding from-scratch restructuring of datasets.

Prediction of concept drift may pose new opportunities for data publishers, increasing the efficiency of their curation pipelines, reducing curation time and allowing them to prioritize maintenance on drifting concepts. Additionally, it can be used to quantify the stability of a dataset or, alternatively, the "unexpected" changes across versions. This is key for the fundamental issue of *preservation of Linked Data*, which aims at archiving of Linked Data and suggests that stable data should be archived first. Our method can be used to measure such stability.

Multiple challenges are open for the future, and we plan to extend this work in several aspects. First, how alternative definitions of concept drift affect the prediction of further drifts poses an interesting question. Second, we plan to apply our approach in other domains to continue investigating its genericity. Third, we intend to use a higher number of versions in order to confirm the trends we observed and to investigate optimal window sizes. Finally, we envisage an extension of our approach in a big data environment, in order to cope with larger datasets and detect change of meaning of concepts in real time.

Acknowledgements The work on which this paper is based has been supported by the Computational Humanities Programme of the Royal Netherlands Academy of Arts and Sciences. For further information, see `http://ehumanities.nl`. This work has been supported by the Dutch national program COMMIT. We acknowledge suggestions contributed by colleagues, especially Sarven Capadisli.

References

1. A. Mäedche, B. Motik and L. Stojanovic: Managing multiple and distributed ontologies in the Semantic Web. VLCB Journal 12(4), 286–300 (2003)

2. Capadisli, S.: Linked Statistical Data Analysis. In: 1st International Workshop on Semantic Statistics (SemStats 2013), ISWC. CEUR (2013)
3. Fanizzi, N., d'Amato, C., Esposito, F.: Conceptual Clustering: Concept Formation, Drift and Novelty Detection. In: The Semantic Web: Research and Applications, 5th European Semantic Web Conference. LNCS 5021. pp. 318–332. Springer (2008)
4. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: classification and survey. The Knowledge Engineering Review 23(2), 117–152 (2008)
5. Gama, J., Medas, P., Castillo, G., Rodrigues, P.P.: Learning With Drift Detection. In: Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence. LNCS 3171. vol. 3171, pp. 286–295. Springer (2004)
6. Gonçalves, R.S., Parsia, B., Sattler, U.: Analysing Multiple Versions of an Ontology : A Study of the NCI Thesaurus. In: 24th International Workshop on Description Logics (DL 2011). vol. 745. CEUR (2011)
7. Gulla, J.A., Solskinnsbakk, G., Myrseth, P., Haderlein, V., Cerrato, O.: Semantic Drift in Ontologies. In: Proceedings of the 6th International Conference on Web Information Systems and Technologies. vol. 2. INSTICC Press (2010)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. Newsl. 11(1), 10–18 (Nov 2009), `http://doi.acm.org/10.1145/1656274.1656278`
9. Kira, K., Rendell, L.A.: The Feature Selection Problem: Traditional Methods and a New Algorithm. In: Proceedings of the Tenth National Conference on Artificial Intelligence. pp. 129–134. AAAI'92, AAAI Press (1992)
10. Klein, M.: Change Management for Distributed Ontologies. Ph.D. thesis, VU University Amsterdam (2004)
11. Lambert, P., Zijdeman, R., Maas, I., Prandy, K., van Leeuwen, M.: Testing the universality of historical occupational stratification structures across time and space. In: ISA RC 28 Social Stratification and Mobility spring meeting, Nijmegen (2006)
12. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web – Interoperability, Usability, Applicability (2014), `http://www.semantic-web-journal.net/system/files/swj558.pdf`
13. Meroño-Peñuela, A.: Semantic Web for the Humanities. In: The Semantic Web: Semantics and Big Data, 10th European Semantic Web Conference. LNCS 7882. pp. 645–649. Springer (2013)
14. Meroño-Peñuela, A., Guéret, C., Hoekstra, R., Schlobach, S.: Detecting and Reporting Extensional Concept Drift in Statistical Linked Data. In: 1st International Workshop on Semantic Statistics (SemStats 2013), ISWC. CEUR (2013)
15. Pesquita, C., Couto, F.M.: Predicting the Extension of Biomedical Ontologies. PLoS Computational Biology 8(9), e1002630 (2012), doi:10.1371/journal.pcbi.1002630
16. Stojanovic, L.: Methods and Tools for Ontology Evolution. Ph.D. thesis, University of Karlsruhe (2004)
17. Tsymbal, A.: The problem of concept drift: definitions and related work. Tech. Rep. TCD-CS-2004-15, Computer Science Department, Trinity College Dublin (2004)
18. Wang, S., Schlobach, S., Klein, M.C.A.: What Is Concept Drift and How to Measure It? In: Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010. Proceedings. pp. 241–256. Lecutre Notes in Computer Science, 6317, Springer (2010)