Information Sciences 503 (2019) 551–573

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Web genre classification with methods for structured output prediction☆

Gjorgji Madjarov[a], Vedrana Vidulin[b,c,d], Ivica Dimitrovski[a], Dragi Kocev[b,*]

[a] Faculty of Computer Science and Engineering, University Ss. Cyril and Methodius, Skopje, Macedonia
[b] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
[c] Ruđer Bošković Institute, Zagreb, Croatia
[d] Faculty of Information Studies, Novo mesto, Slovenia

ARTICLE INFO

Article history:
Received 1 December 2016
Revised 17 June 2019
Accepted 1 July 2019
Available online 8 July 2019

Keywords:
Web genre classification
Hierarchy construction
Hierarchical multi-label classification

ABSTRACT

The increase of the number of web pages prompts for improvement of the search engines. One such improvement is specifying the desired web genre of the resulting web pages. The prediction of web genres triggers expectations about the type of information contained in a given web page. More specifically, web genres can be seen as textual categories such as scientific papers, home pages or eshops. Arguably, in the context of web search, specifying genre beside topical keywords enables a user to easily find a scientific paper (genre) about text mining (topic). Typically, web genre prediction is treated as a predictive modelling task of multi-class classification, with some recent studies advocating the introduction of a structure in the output space: either by considering multiple web genres per web page or exploiting a hierarchy of web genres. We investigate the structuring of the output space by constructing hierarchies using data-driven methods, experts or even randomly. We also use 10 different representations of the web pages. We use predictive clustering trees and ensembles thereof to properly assess the influence of the different information sources. The experimental evaluation is performed on two benchmark corpora: 20-genre and SANTINIS-ML. The results reveal that exploiting a hierarchy of web genres yields best predictive performance across both datasets, all predictive models, all feature sets and all hierarchies. Next, data-driven hierarchy construction is at least as good as expert-constructed hierarchy with the added value that the hierarchy construction is automatic and fast. Furthermore, ensembles offer state-of-the-art predictive performance and they have a superior performance than single tree models.

1. Introduction

There is an increasing need for new ways of searching for desired web pages on the Internet (in December 2017 there were more than 1.3×10^9 websites – http://www.internetlivestats.com). Typically, searching is performed by typing keywords in a search engine that returns web pages of a topic defined by those keywords. The user could, however, obtain more

☆ The first two authors should be regarded as joint first authors.

* Corresponding author.
E-mail addresses: gjorgji.madjarov@finki.ukim.mk (G. Madjarov), vedrana.vidulin@ijs.si (V. Vidulin), ivica.dimitrovski@finki.ukim.mk (I. Dimitrovski), dragi.kocev@ijs.si (D. Kocev).

https://doi.org/10.1016/j.ins.2019.07.009
0020-0255/

precise results if a web page genre is specified in addition to the keywords and used to filter out irrelevant pages returned by keywords [52]. Furthermore, automatic identification of web page genres could facilitate focused crawling of web pages, e.g., retrieval of research articles, book chapters and theses for plagiarism analysis [46]; web page abstraction, e.g., by knowing specific features of a genre, it is easier to extract only content-related text for a braille reader [46]; or automatic extraction of metadata for management of digital documents in digital libraries [39].

Genre can be defined as "a particular type or style of literature, art, film or music that you can recognize because of its special features" [15]. We follow the genre definition from [44] stating that recognition of document (web page) genre reduces cognitive load by triggering expectations about the type of information contained in the document. Genres are recognizable based on specific conventions/features. "For instance, if we read a sequence of short questions and brief answers (conventions), we might surmise that we are reading FAQs (genre); we then realize that the purpose of the document is to instruct or inform us (expectations) about a particular topic or event of interest." [44]. In the context of web search, specifying genre beside keywords enables a user to easily find a scientific paper (genre) about the topic of text mining.

A web page is a complex document that can share conventions of several genres or contain parts from different genres. While this is recognized in the web genre classification community, state-of-the-art genre classifier implementations still attribute a single genre to a web page from a set of predefined genre labels (i.e., address the task as multi-class classification). However, a line of research [41,44,53] advocates that multi-label classification (MLC) scheme is more suitable for capturing the web page complexity. The rationale is that since several genres are easily combined in a single web page, such hybrid forms thus require attribution of multiple genre labels. For example, a web page of the genre 'Blog' can share conventions of 'Personal', 'Journalistic' or 'Informative' genres. A 'Blog'-'Personal' web page is a blog that expresses a personal opinion of blog's author, e.g., a personal experience on using an application (Fig. 7). A 'Blog'-'Journalistic' page represents a blog with information of a current interest to the reader, e.g., a blog with news on weather conditions (Fig. 8). A 'Blog'-'Informative' web page represents a blog that reports on information interesting for a longer period of time, e.g. a computer programming how-to (Fig. 9). Furthermore, web genres naturally form a hierarchy of genres. For example, 'Personal home page' is a type of 'Personal'.

The aforementioned properties of the web genre classification can be easily mapped to the machine learning task of hierarchical multi-label classification (HMC). HMC is a variant of classification, where a single example may belong to multiple classes at the same time and the classes are organized in the form of a hierarchy. An example that belongs to a class automatically belongs to all its super-classes – the hierarchical constraint. Problems of this kind can be found in many domains including text classification, functional genomics, and object/scene classification.

Although it can be easily conceived that the task of web genre classification can be mapped to HMC, the hierarchical and multi-label structure of web genres has not yet been explored. There are three major obstacles for this. First, there is a lack of a comprehensive genre taxonomy with a controlled vocabulary and meaningful relations between genres and web-page-based corpora labelled with such a taxonomy [9]. Second, from a machine learning point of view, methods that are able to fully exploit the complexity of such data started appearing only recently and have not yet gained much visibility [23,45]. Finally, the deficiency of available resources for web genre classification heavily impedes the bridging of the machine learning and web genre research communities.

In this work, we aim to address all of these obstacles. First of all, we propose structuring of web genres into hierarchies by an expert and propose to use methods for generating hierarchies using the available data. The use of data-driven methods would bypass the complex process of hierarchy construction by experts: it is difficult (if at all possible) to construct a single hierarchy that would be acceptable by all of the experts. Second, we take two benchmark corpora for genre classification (*20-Genre* [53] and *SANTINIS-ML* [18,44]) and convert them into HMC datasets. Furthermore, we investigate the influence of the hierarchy of web genres on the predictive performance of the predictive models. For this purpose, we define four machine learning tasks that could be used in the context of web genre prediction: single-label classification (SLC, each web page is annotated with single genre), multi-label classification (MLC, each web page is annotated with multiple genres), hierarchical single-label classification (HSC, each page is annotated with a single genre and the genres are organized into a hierarchy) and hierarchical multi-label classification (HMC, each page is annotated with multiple genres organized into a genre hierarchy) [29]. Finally, we made a repository of all the data used in this study to be publicly available for research purposes (http://webgenres.ijs.si).

We also investigate the performance of several feature extraction techniques used for representing web pages. Namely, we use features based on linguistic properties of the web pages (surface, structural, presentation, context), feature types using neural network models and a feature type based on character *n*-grams.

For accurately measuring the contribution of the hierarchy and reducing the model bias, we need to consider a predictive modelling method that is able to construct models for all of the four aforementioned machine learning tasks. Such a methodology is offered with the predictive clustering trees (PCTs) [23]. PCTs can be seen as a generalization of decision trees for the task of predicting structured outputs, including the considered tasks of SLC, MLC, HSC and HMC. Moreover, we construct ensembles of PCTs (bagging [5] and random forests [6]) – state-of-the-art predictive models. By doing so, we investigate the influence of the structure of the web genres to the predictive performance of the ensembles of PCTs.

An initial investigation of web genre classification with methods for structured output prediction is given in [32]. We extend this work along several dimensions. First of all, we have introduced an additional multi-label benchmark corpus (i.e., we added the *SANTINIS-ML* corpus). Second, we consider two additional machine learning tasks: SLC and HSC. Third, we

have added two additional types of features based on neural network models and *n*-grams. Furthermore, we have updated the pipeline for extraction of linguistic genre-specific features. Next, we use additional methods for constructing a hierarchy of web genres. Moreover, we push this to an extreme and consider random hierarchy construction. Finally, we make all of our data public for further use by the machine learning and web genre research communities.

The remainder of this paper is organized as follows. Section 2 gives the motivation and the background for the work presented here. The web genres corpora and the feature extraction techniques are discussed in Section 3. Section 4 outlines methods for inducing hierarchies of web genres, while Section 5 explains the predictive modelling methodology used here, i.e., predictive clustering trees. The experimental design is described in Section 6, while the results are discussed in Section 7. Finally, Section 8 concludes and provides directions for further work.

2. Motivation and background

2.1. Usefulness of automatic web genre identification in search engines

The claim that identifying genres of web pages returned by topic-based search would help to improve precision of search results was first made by Karlgren and Cutting [21]. They constructed a multi-class genre classifier based on 15 genres from the Brown corpus and use discriminant analysis aiming to *"take a set of texts that has been selected by some sort of crude semantic analysis such as is typically performed by an information retrieval system and partition it further by genre or text type, and to display this variation as simply as possible in one or two dimensions"*[21]. Furthermore, an information retrieval application is discussed where newsreader system would use the genre classifier on USENET news texts, previously sorted by topics. Genres in this application are: "query", "comment", "announcement", "FAQ", etc. The main focus is then on the components of the web genre classifier and not on measuring the usefulness of the classifier in the context of a search engine.

Rosso [40] performed a user study to explore whether users perceive genres as useful in a web search context. Participants were presented with multiple tasks, e.g., to find web pages containing information on courses that cover introductory writing skills. For each task, they were then presented with a summary search result with a list of links to a set of web pages for which they needed to assess the relevance regarding the task. The summaries also contained information on web page types, i.e., genres. The results of the study showed that only half of the participants actually used genre labels when making decision on a web page relevance. When explaining how they used genres, the predominant argument was that genres helped them to filter out those pages that were not interesting to them, e.g., to filter out blogs as less credible source of information. At the first sight, the study shows partial usefulness of genre labels in detecting web page relevance. However, the limitation of the study is that the conditions in which the participants assessed the relevance were different than the typical web search behavior. Many participants stated that the topics of the presented tasks were not very familiar to them. Consequently, their focus could be shifted towards understanding the topic, which might explain why many of them ignored the genre labels.

Vidulin et al. [52] performed a web page retrieval experiment that compared the relevance of results obtained by keyword-based vs. keyword-and-genre-based search. Pages from the 20-genre corpus were indexed with a desktop version of Google. The first task was to examine whether it is possible to obtain a web page of desired topic and genre solely through keyword-search. Consequently, 18 sets of keywords were inserted into the search engine each containing a hint of genre for which a user is looking for, e.g., "madonna lyrics", "health research". In the second phase, genre label was used beside the same sets of keywords to constrain the search. The relevance was measured by using the evaluation measure precision at 10, which is the fraction of relevant hits among the top 10 hits. The results showed that by constraining the search with genres the precision has doubled. The number of hits was also reduced, but this is not so critical for search engine users. If a user has a very specific query and the total number of relevant hits is consequently low, a search engine can automatically ignore the genre.

Stein et al. [46] analyzed conditions that need to be satisfied for including a web genre classifier into a search engine. They argue that a web genre classifier should be based on a feature set that is computationally inexpensive and exportable to corpus other than the one on which a classifier is constructed. Next, the proposed feature set is composed of less than 100 features including HTML tags, links, characters and words obtained through genre core vocabulary analysis, and showed that it satisfied both criteria. The classifier based on these features, as well as a linear discriminant analysis were incorporated into a Firefox plug-in named WEGA. The plug-in added genre labels to the snippets presented by a search engine as a result of a topic-based keyword search.

Based on the range of applications for which automatic identification of web page genres can be helpful [8,25,33,39,46,52], we consider web genre classifier as a useful addition to search engines. The focus of this paper is on improving the predictive performance of the classifier by automatic structuring of genre labels and using machine learning algorithms that can exploit this structure during learning of the predictive models. We expect that the improved performance will increase the number of web pages for which we can successfully predict genres (from the existing genre palette), consequently helping information retrieval applications to increase the precision of search results.

2.2. Machine learning tasks

From a machine learning perspective, the web genre prediction can be represented with four different tasks exploiting different aspects of the output space. To begin with, the most simple task is single-label classification (SLC) where each page is annotated with a single web genre. Second, multi-label classification (MLC) allows a single web page to be annotated with multiple web genres. Next, hierarchical single-label classification (HSC) imposes a hierarchy over the web genres and exploits the hierarchical information - each web genre is annotated with a single web genre from the bottom level of the genre hierarchy. Finally, hierarchical multi-label classification (HMC) allows for multiple annotations per web page and the genres are organized into a hierarchy.

It has already been pointed out that web pages do not easily fall into a single genre [53]. Many web pages are hybrid [42] and users often disagree when deciding the genre of a web page [43]. The tasks described above aim to address the web genre classification considering different complexity aspects of the web genres output structure. First, let us consider web genre classification as a SLC task where we learn a separate predictive model for each web genre. A page is then classified by applying all of the separate predictive models and use the obtained predictions to obtain the set of web genres that the given page is annotated with. Conversely, from the point of view of MLC, we learn a single predictive model for all web genres. A page in this case is classified by applying this model and use its predictions to annotate the given page. If we impose a hierarchical structure on the web genres, then the respective predictive models for HSC and HMC behave similarly as their non-hierarchical counterparts. The former type of predictive models is called local, while the latter group global models [23,45].

The global methods have several advantages over the local methods. First, they exploit and use the dependencies that may exist between the components of the structured output in the model learning phase, which can result in better predictive performance of the learned models. Next, they are typically more efficient: it can easily happen that the number of components in the output is very large (e.g., hierarchies in functional genomics can have several thousands of components), hence learning a model for each component is not feasible. Furthermore, they produce models that are smaller than the sum of the sizes of the models built for each of the components.

Formally, the four tasks can be defined as follows [29]. The input space $\mathcal{X}$ consists of tuples of values of primitive data types (boolean, discrete or continuous), i.e., $\forall \mathbf{x_i} \in \mathcal{X}, \mathbf{x_i} = (x_{i_1}, x_{i_2}, \ldots, x_{i_D})$, where D is the size of the tuple (or number of descriptive attributes). These are the features describing a web page.

The output space $\mathcal{Y}$ is defined based on the level of structure in the output space. Let $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_Q\}$ is the set of Q possible labels. In SLC, the output space is defined as $\mathcal{Y} = \mathcal{L}$, i.e., a single value from the set of values and $Q \geq 1$. Next, in MLC, the output space is $\mathcal{Y} = 2^{\mathcal{L}}$ with $Q > 1$ and $\mathcal{Y}_\mathbf{i} \subseteq \mathcal{L}$, i.e., all possible subsets of labels including the empty set. Finally, in HSC and HMC, the output space is defined with a label hierarchy $(\mathcal{L}, \leq_h)$, where $\mathcal{L}$ is a set of labels and $\leq_h$ is a partial order representing the ancestor-descendant relationship ($\forall\ \lambda_1, \lambda_2 \in \mathcal{L} : \lambda_1 \leq_h \lambda_2$ if and only if λ_1 is an ancestor of λ_2) structured as a tree [23]. Moreover, in HSC, each example is annotated with a single label (i.e., web genre), while in HMC each example can be annotated with multiple web genres.

We next describe the set of examples E as pairs of a tuple and a label set from the input and output space respectively, i.e., $E = \{(\mathbf{x_i}, \mathcal{Y}_\mathbf{i}) | \mathbf{x_i} \in \mathcal{X}, \mathcal{Y}_\mathbf{i} \subseteq \mathcal{L}, 1 \leq \mathbf{i} \leq N\}$ where N is the number of examples of E ($N = |E| = |\mathcal{X}|$). Finally, we define a quality criterion q that rewards models with high predictive performance and low computational complexity. Considering all of the above, the goal is then to find a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that maximizes the quality criterion q.

Finally, note that one can address multi-label classification by decomposition of the labels into several subsets of labels and then apply multi-label learning methods on the subsets [7,19,29,50]. Mainly these exploit the decompositions to create ensembles of predictive models whose predictions are then combined to obtain the prediction for the complete label space. Moreover, the obtained subsets of labels are typically overlapping. However, there are some remaining issues mainly connected with the clustering requiring further attention before this could be directly applied in our setting

2.3. Bridging two research communities

State-of-the-art approaches for web genre classification mostly deal with feature construction and use the benchmark *7-Web* and *KI-04* multi-class corpora to evaluate the quality of the obtained feature sets [44]. These two corpora focus on a set of web genres that are on a same level of a hierarchy – experimental work has indicated that a mix of genres from different levels may considerably deteriorate the predictive performance of a multi-class classifier [41]. In a MLC setting, typically used corpus is the *20-Genre* corpus [53]. The MLC task is addressed by using binary relevance where for each genre a separate classifier is constructed, e.g., by computing a genre-specific centroid from the data [17] or by hand-crafting genre-specific rules [44]. Since the relations between genres are not explicitly considered in the classifier construction phase, the presented classifiers often exhibit either high precision and low recall or vice versa.

A hierarchical (although not multi-label) corpus is presented in [47]: An expert constructed a two-level tree-graph hierarchy composed of 7 top-level and 32 leaf nodes. The authors then hand-crafted rules separately for each leaf node arguing that 40 examples per leaf are not sufficient for data-driven classifier construction. The potential relations between genres were explicitly encoded in the form of a dependency graph ordering 32 classifiers as follows. If a classifier for genre Y tends to mistakenly classify genre Y as X, then in the sequence of classifiers the classifier for genre Y is ordered behind the clas-

sifier for genre X. A new web page is classified into the genre of the first classifier in the sequence that returns a positive answer.

Besides the aforementioned hierarchical single-label corpus [47], Wu et al. [54] used three other hierarchical, but single-label corpora with genres organized into a hierarchy of up to three levels of depth: Brown [24], BNC [28] and Syracuse [9]. Brown corpus has 500 documents labeled with one of the 15 bottom level genres. In the BNC corpus 4053 documents are annotated with 70 bottom level genres, and in the Syracuse 3027 web pages with 292 genres ([54] used only genres containing 15 or more examples, which resulted in 2293 pages annotated with 52 genres). Wu et al. [54] measured the difference in predictive performance between a SVM classifier with flat labels and a structural SVM classifier with structured labels. Hierarchy-based approach remarkably outperformed flat label approach only on one (Brown corpus) out of four corpora. It should be noted that Brown and BNC are not composed of web pages, which eliminates the use of web page specific features such as html tags and links. These are computationally inexpensive features that considerably contribute to web genre classifier's predictive performance [20,46]. Similar problem is with KRYS I corpus [3], which is composed of 6300 PDF documents classified into one of the 70 genres connected in a hierarchy with 10 super-genres.

A recent initiative to construct a large web genre benchmark corpus is based on crowdsourcing web genre annotations [1]. The resulting Leeds Web Genre Corpus is constructed in two steps. In the first step, 3964 web pages were obtained through focused search. Each web page was annotated by five annotators through the Mechanical Turk web site. To keep the annotation task simple, the authors opted for single-genre/single-label annotation approach. The quality of this part of the corpus is reflected in high inter-annotator agreement (for 95% of the web pages at least four annotators agreed on a label). In the second step, the corpus was extended by collecting a random sample of 1000 web pages from the Internet. Besides 15 genres, five additional genres were detected in the new sample. Annotations for the new sample were also crowdsourced. While the corpus is of high quality, it is not suitable for our proposed approach for automatic construction of web genre hierarchies, since the corpus contains single-genre-per-page annotations while we focus on corpus with multiple-genres-per-page annotations.

An overview of state-of-the-art approaches for web genre classification indicates a lack of data that captures both hierarchical and multi-label aspects of web genres. The lack of data hinders the development of data-driven hierarchical multi-label web genre classification tools. Furthermore, the public availability of web genre corpora is limited even for simple single-label web genre classification tasks. The available corpora is scattered through several web sites. We contacted many authors of publications dealing with web genre classification but we failed to obtain additional corpora to the ones that are already available. Furthermore, to the best of our knowledge, none of the authors published the constructed data sets, i.e., datasets that can be readily used by machine learning methods. This raises a need for a web genre data repository.

One of the aims of this paper is to bridge the web genre and machine learning communities. The collaboration between these two communities was thus far impeded by the lack of available data. While we do not have the rights to publish the corpora of other authors, we can provide the machine learning community with a repository of data sets covering several distinct feature sets computed from the available corpora. In addition, the web genre community will be provided with several of the interpretable feature sets, which can be used for investigating features that best describe each genre or groups of genres. More specifically, we provide the web genre community with a machine learning methodology able to grasp structured data (i.e., methods for HMC, MLC, HSC) and guidelines for annotation of future web genre datasets, e.g., when is feasible to introduce a structure of the output space or which feature set is the most genre-distinctive in combination with a specific classification method. Finally, we provide the machine learning community with pre-computed data sets that are ready for use thus facilitating the development of novel web genre classification methods.

3. Web genres data

3.1. Corpora description

The **20-Genre Collection corpus** [53] was composed with an aim to construct a web genre classifier that would annotate web pages with genres within a search engine. Consequently, selected genres mostly form broad categories (e.g., *Informative*) with addition of some specific genres that are of high interest for a user (e.g., *FAQ*) and uninteresting genres that the user would like to filter out (e.g., *Error message*). The corpus is composed of 1539 web pages in English. Pages were collected by first using highly-ranked Google hits for popular keywords from a Year-End Google statistics. Such pages should improve classifiers capabilities to annotate pages that are actually searched for. Random pages were then searched for by using Mangle (http://www.mangle.ca/), a random link generator. Finally, pages belonging to the under-represented genres were specifically searched for by inputting genre-related queries into Google. Collected pages were annotated with several of the 20 genres (Table 1) by two independent annotators. The labels were then further assessed by two additional annotators. The label cardinality (average number of labels per example) is 1.34 and the label density 0.067 (label cardinality divided by the number of labels).

SANTINIS-ML is a subset of SANTINIS corpus [44] with multi-label (-ML) genre annotations. The subset is originally composed of 1000 English web pages from the SPIRIT collection [18] manually annotated with 15 genres from the rest of the SANTINIS corpus [44]: 7 web genres from the 7-webgenre collection, 4 traditional BBC web genres and 4 rhetorical genres (the last four genres in the Table 2; they form broad categories that intertwine with the rest of the web genres). The pages were annotated by Santini [44] with zero (pages that could not be annotated with any of the 15 genres), one

Table 1
Web genres in 20-Genre collection corpus.

Genres	#	Description
Blog	77	Blogs, diaries, time-stamped updates
Childrens'	105	Pages presented in a simple and colorful way intended for children, e.g., encyclopedia and lyrics for children
Commercial/ promotional	121	Homepages of institutions, organizations, political parties, institutionalized individuals; product descriptions; service descriptions; press releases
Community	82	Forums, news group pages, portals with user-generated content
Content delivery	138	Download pages, image and movie galleries, games
Entertainment	76	Jokes, puzzles, horoscopes, games
Error message	79	Custom HTTP error pages, non-HTTP errors
FAQ	70	Frequently asked questions
Gateway	77	Introductory pages, redirection pages, login pages
Index	227	Link collections, table of contents
Informative	225	Encyclopedic materials, recipes, user manuals, how-tos, lecture notes for a wide audience, informative books, biographies, discographies, filmographies
Journalistic	186	News, reportages, editorials, interviews, reviews
Official	55	Legal materials, official reports, rules
Personal	113	Personal homepages, pages with opinions, descriptions of interests and activities
Poetry	72	Poems, lyrics
Pornographic	68	Pages for adult containing pornographic pictures, videos and stories
Prose fiction	67	Fanfiction story, short story, novel
Scientific	76	Papers, theses, lecture notes for a specialized audience, scientific books
Shopping	66	Online stores, classified ads, price comparators, price lists
User input	84	Forms, surveys

Table 2
Web genres in SANTINIS-ML corpus.

Genres	#	Description
E-shop	27	Pages that sell products
FAQ	13	Frequently asked questions
Front page	8	The first page of a newspaper
Listing	391	Hotlists, sitemaps, tables of contents, checklists
Personal home page	21	Pages in which a person presents himself/herself
Search page	76	Dynamic page focused on presenting results of a search
Editorial	6	An opinion written by a newspaper editor, which typically represents the opinion of the whole newspaper
DIY mini-guide	4	Pages with a list of steps, description of tools and time needed to complete a project
Short biography	14	A detailed description of a person's life
Feature	4	Newspaper pages representing a specific theme
Argumentative-persuasive	123	Pages focused on convincing using logical or emotional arguments
Descriptive-narrative	159	Pages that describe location and time of events
Expository-informational	195	Pages that convey information
Instructional	68	Pages describing activities

or more (up to 4 genres) of the 15 genres. While the disadvantage of the SANTINIS-ML corpus is that it is annotated by a single annotator, it is only multi-label corpus besides 20-genre collection as far as we know. From the 1000 pages, we kept for analysis 705 pages that received at least one label. We removed the *Blog* genre because it was represented with only one page. The label cardinality is 1.57 and the label density is 0.105.

3.2. Linguistic and presentational features for web genres

We used the pipeline presented in Fig. 1 to obtain four types of features for web genres: surface, structural, presentation and context features. Note that this pipeline has been updated from the one used in [32,53] as follows. First, we used a up-to-date linguistic annotation software: ANNIE plug-in from GATE v8.1 [10]. Second, we constructed feature extractors that automatically construct feature sets and compute feature values from GATE XML files with annotations. The new pipeline also applies improved feature value normalization that emphasizes distinctions between rarely and frequently occurring features. Finally, the pipeline is general enough to be applied to any web page corpus, hence facilitating work with future possible web genre corpora, and is modular such that, for example, the context extractor may be left out in absence of URL information.

Surface features represent the web page content and enable extraction of word and symbol occurrence patterns in web genres. For example, the words 'blog' and 'post' occur in Blog pages or the function word 'you' frequently occurs in Commercial/promotional pages. This feature set includes: function words; genre-specific words; punctuation marks; classes of words: person, location, organization, money, percent, date and address; and text statistics: average number of characters

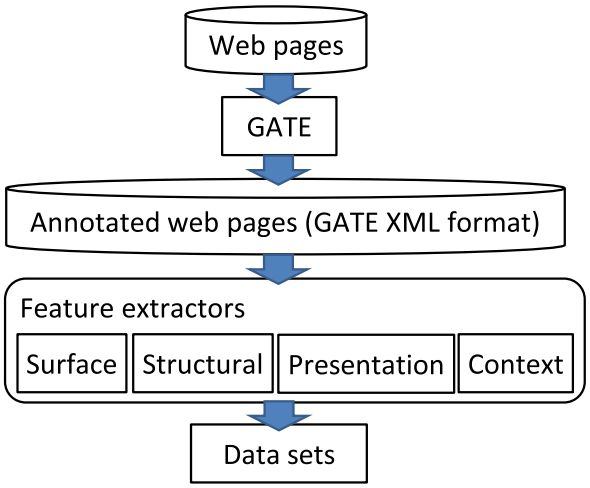

Fig. 1. Web genre-specific feature extraction pipeline exploiting some linguistic properties of the text from the web pages.

per word, median of the number of words per sentence, proportion of hyperlinked text and page text length. The first three feature groups are automatically extracted from the pages by first extracting all possible features that belong to a specific group (e.g., function words) and then by reducing the features to a manageable level by keeping those that appear in at least 1% of the corpus pages. Higher percentage may remove features informative for poorly represented genres (such as *Front page* in SANTINIS-ML). A set of genre-specific words is further pruned by stemming the words using the Porter stemming algorithm [37] and fusing words with the same stem into a single feature.

Feature values mostly represent frequencies of features within web pages (except text statistics) normalized to emphasize distinctions between rarely and frequently occurring features:

$$nf(i, G) = 0.1 + 0.9 \cdot \frac{f_{i,G}}{\max\{f_{i',G} : i' \in G\},} \tag{1}$$

where *nf* is the normalized frequency, *i* is a feature (e.g., word 'you') in a feature group *G* (e.g., function words), *f* is frequency of *i* within a page, which is divided by the maximum frequency observed between features in *G*, and weights are selected to emphasize distinction between rarely and frequently occurring features in *G*.

For genre-specific words, we used two variants of these features: bag-of-words (*BOWSrf*) and term frequency-inverse document frequency (*TFIDFSrf*). The former represents presence/absence patterns of words in pages, while the latter reduces feature importance (*nf*) with the increase in the number of corpus pages (N) in which the word i occurs (n_i):

$$\text{tf-idf}(i, G) = \text{nf}(i, G) \cdot \log_2 \frac{N}{n_i} \tag{2}$$

Structural features (*Struc*) capture syntactic patterns indicating that high frequency of nouns is related to Informative and verbs to Instructional pages. These features include: parts of speech (POS) tags; POS trigrams; and sentence types: declarative, interrogative, exclamatory and other (captions, list items and similar). We consider 35 POS tags from Hepples' tag set [14] obtained by removing tags with unknown usage (identifed with GATE) and tags representing symbols, which are part of other feature set. POS trigrams are intended for capturing occurrences of syntactic structures such as subordinating relations in the text. To reduce the number of possible trigrams, we use sentence segmentation information and extract only those combinations of tags that appear within a sentence. In the cases of both single POS and POS trigrams, we keep only the features appearing in at least 1% of the corpus pages and compute *nf* within both feature groups. The feature values for sentence types represent proportions of types within pages.

Presentation features (*Pres*) represent the formatting of a page and may indicate that html tags such as "form" and "input" are related to User input pages. These features include: proportions of page content covered by different tokens (word, number, punctuation mark and symbol, and space tokens), *nf* of individual html tags, and five tag categories measuring the proportions of tags belonging to each of the five categories. Tag categories represent an amount of text formatting, document structuring, inclusion of external objects, interaction and navigation applied to a page [41] (see Table 3).

Context features (*Context*) describe web page context in terms of page URL (e.g., appearance of word "archive" in Blog pages) and types of hyperlinks within the page (e.g., high proportion of hyperlinks to a different domain in Index pages). These features include: URL depth (the number of directories in URL path); whether the page is static or dynamic; presence of the following elements in the URL: https, tilde, each of the top level domains (com, org, edu, net, gov, mil, int) or national domain, www, year, query, fragment and words contained within the URL (stemmed; words that appear in at least 1% of corpus pages URLs); proportions of hyperlinks to the same domain, to a different domain and mailto hyperlinks.

Table 3
Five HTML tag categories and tags they account for.

HTML tag group	Tags
Text formatting	abbr, acronym, address, b, basefont, bdo, big, blockquote, center, cite, code, del, dfn, em, font, h1, h2, h3, h4, h5, h6, i, ins, kbd, pre, q, s, samp, small, strike, strong, style, sub, sup, tt, u, var
Document structure	br, caption, col, colgroup, dd, dir, div, dl, dt, frame, hr, iframe, li, menu, noframes, ol, p, span, table, tbody, td, tfoot, th, thead, tr, ul
Inclusion of external objects	applet, embed, img, object, param, script, noscript
Interaction	button, fieldset, form, input, isindex, label, legend, optgroup, option, select, textarea
Navigation	a, area, link, base

Summary for the linguistic features. From the *20-Genre collection* corpus, we constructed four data sets composed of 5698 surface, 5682 structural, 69 presentation and 38 context features. Similarly, from the *SANTINIS-ML* corpus, the four datasets consist of 3446 surface, 3384, structural, 68 presentation and 40 context features.

3.3. Paragraph vector features

Paragraph Vector [27] (PV) features are continuous distributed vector representations for variable-length texts, ranging from sentences and paragraphs to documents. These features are inspired by the research in learning vector representations of words using neural networks. There are two variants of the PV features: a Distributed Memory (DM) model and a Distributed Bag of Words (DBOW) model. In this work, we specifically investigate their effectiveness on representing web pages for web genre classification.

The PV model builds on previous methods for generating distributed representations for words also known as word embeddings [35,36]. The main idea is that the syntactic and semantic meaning of a word can be inferred from accompanying words given large enough text corpus. By training the model to predict the next word given a sequence of words, a suitable representation for the word is generated. The DM model works similarly with the only difference being that the paragraph vectors are also used to generate the next word prediction.

The model can be seen as having a memory component represented by a paragraph vector which stores information about the content of a given document. Paragraph vectors are shared across all possible context windows in the current paragraph, but are not used for the other paragraphs. The word vectors learned during the training phase, on other hand, are shared for all paragraphs. Due to the memory complexity of the PV implementation, we only utilize vector summing and not vector concatenation when generating the next word prediction. In this way, we lose some of the word ordering information, but we still get satisfactory representations.

The DBOW model does not consider the word ordering nor does it generate representations for the words. What the model does is, at each iteration it samples a text window and a random word from the current context. It then forms a classification task given the paragraph vector. It works conceptually similar to the Skip-gram model [36].

Le and Mikolov [27] suggest a joint representation of the two models. Consequently, for each page, we train a DM and a DBOW model with an equal paragraph vector size. The final representation is a combination of the output from the two models. We parametrized the algorithm as follows. First, for both models, we set a context window size of 10 words. Next, the learning rate is set to 0.025 and is decreased by 0.002 after two epochs. Furthermore, we generate document representations with a dimensionality of 600 features. Finally, these feature representations are generated for both the raw web pages (i.e., HTML documents; *PV_R*) and cleaned version with HTML tags removed (only the actual content is used; *PV_C*).

We also use a collection of pre-trained word vectors provided by the GloVe algorithm [16]. The pre-training is performed on word-word co-occurrence statistics from a Wikipedia 2014 corpus, where the obtained representations are interesting linear substructures of the word vector space. The pre-trained models are then fine tuned on our two corpora. Similarly as the other PV, we used representations with 600 features. Depending on whether the fine tuning is performed on raw or cleaned web pages (removal of the HTML tags), we obtained two feature sets: *PV_Gl_R* and *PV_Gl_C*, respectively.

3.4. N-gram features

The *n*-gram features were first proposed in [20]. There were several different features for representing web pages proposed: character *n*-grams and word *n*-grams, both coupled with binary and term-frequency representation. Furthermore, they explore the effectiveness of adding structural information. An experimental evaluation on corpora with single label web genres shows that binary character *n*-grams are the most effective standalone features [20]. Consequently, these are the features used here.

We use *n*-grams with size of 3, 4 and 5 and we compute the frequency of each of the generated *n*-grams. The initial feature sets consist of three equal size subsets of the most frequent 3-grams, 4-grams and 5-grams. We consider the 8000 most common *n*-grams. We then apply feature selection to reduce this number. The importance of a *n*-gram (C) is represented by a "glue" function $g(C)$. For each C, we compute the glue of its antecedent and successor. These are strings consisting of $n-1$

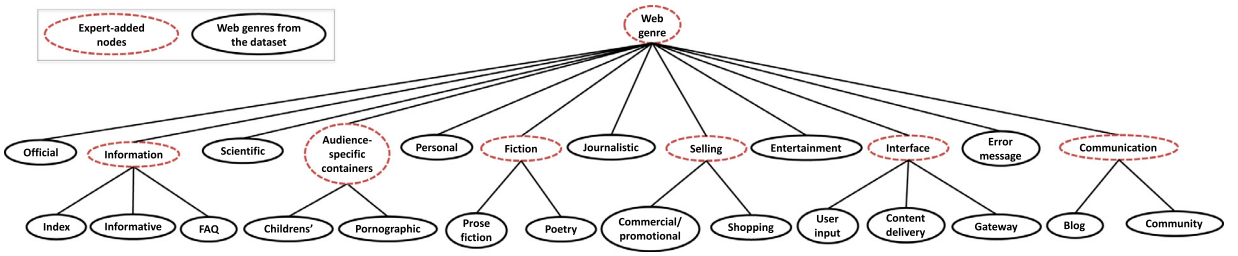

Fig. 2. Web genre hierarchy constructed by an expert for the 20 Genre dataset.

consecutive characters of C and string of size $n+1$ with one extra character on the left and right respectively. A n-gram is considered dominant if both its successor and antecedent have smaller glue value. Exceptions are made for 3-grams which are not compared with their antecedents and 5-grams to their successors.

The glue value is computed using a symmetrical conditional probability [11]: a product of the conditional probabilities calculated as $SCP(x, y) = p(x \mid y) \cdot p(y \mid x)$. A given n–gram can be divided into two subparts at a 'dispersion point'. An n–gram contains $n-1$ possible dispersion points. These dispersion points are used to see if the character n-gram is more important than the two substrings defined by the dispersion point. Finally, the glue is computed by measuring the so called *fairSCP*, which takes into account all potential dispersion points:

$$fairSCP(c_1 \dots c_n) = \frac{p(c_1 \dots c_n)^2}{\frac{1}{n-1} \sum_{j=1}^{j=n-1} p(c_1 \dots c_j) p(c_{j+1} \dots c_n)} \tag{3}$$

Using the feature selection procedure, from the 8000 most common n-grams we selected 7262 and 7345 n-grams (*NGrams*8000) with binary representation for the *20-Genre collection* and *SANTINIS-ML* datasets, respectively.

4. Structuring the output space

There are two major approaches to construct hierarchy of web genres: expert-driven and data-driven. The former approach is based on expert knowledge of the different web genres, while the latter uses the available web genre annotations (i.e., genre co-occurrences) to induce a hierarchy.

4.1. Expert-driven hierarchy construction

The expert-driven hierarchy construction was performed under two constraints: the genre labels need to be the leaves of the hierarchy and the hierarchy needs to be tree-shaped. These two constraints come from the specifics of the predictive modelling methodology at hand. In this way, each original genre label is a part of only one group of labels. This imposes certain limitations on the degrees of freedom while constructing the hierarchy. For example, in the case of SANTINIS-ML genres, both Eshop and Editorial are of Argumentative-Persuasive type. Eshops try to persuade potential buyers to buy a certain product or service [41], while editorials are argumentative statements of views that are considered to be representative of a newspaper as a whole [41]. However, a choice has been made to group Editorial with Front page and Feature under the meta-category Journalistic, which is also present in the 20-genre collection. Since not all members of the Journalistic meta-category are of Argumentative-Persuasive type, this label has been grouped with Eshops.

Construction of expert-based hierarchy for the 20-Genre Collection was guided by definitions of genres from [39] presenting the results of experts efforts to construct a unified web genre hierarchy. For the SANTINIS-ML collection, we used the provided annotations and their descriptions [44] to find meaningful relations between genres, e.g., DIY provides information in the form of instruction on how to complete a project; therefore, we grouped nodes DIY and Instructional under the common meta-category Information. The constructed hierarchies for the 20-Genre Collection and the SANTINIS-ML collection are depicted in Figs. 2 and 3, respectively.

4.2. Data-driven hierarchy construction

When we build the hierarchy over the label space, there is only one constraint that we should take care of: the original MLC task should be defined by the leaves of the label hierarchy. In particular, the labels from the original MLC problem represent the leaves of the tree hierarchy, while the labels that represent the internal nodes of the tree hierarchy are meta-labels (that model the correlation among the original labels). An investigation of the use of label hierarchies in MLC constructed in a data-driven manner shows that it improves the predictive performance[29,30].

In this study, we consider flat MLC label-sets and construct label hierarchies from the label sets that appear in the annotations of the training data by using four clustering methods (the first two are divisive and the second two are agglomerative):

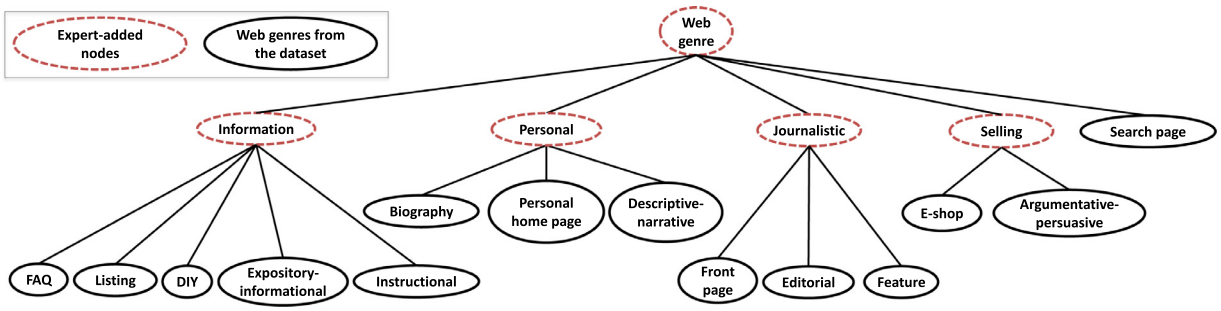

Fig. 3. Web genre hierarchy constructed by an expert for the SANTINIS-ML dataset.

- balanced *k*-means clustering (BkM) [49],
- predictive clustering trees (PCTs) [4],
- clustering with complete linkage (CL) [34], and
- clustering with single linkage (SL) [34].

Balanced *k*-means creates the label hierarchy by partitioning the original labels recursively in a top-down depth-first fashion. The top node of the hierarchy contains all labels. At each node n, $k <= |\mathcal{L}_n|$ child nodes are created. The labels of the current node are distributed (divided) using a clustering method into k disjoint subsets (k meta-labels) with an explicit constraint on the size of each subset, one for each child of the current node.

In this work, we use a specific setting from the predictive clustering framework [12,22], where the target space is the same as the descriptive space, i.e., the descriptive variables are used to provide descriptions for the obtained clusters. This focuses the predictive clustering setting on the task of clustering instead of prediction. Agglomerative clustering algorithms treat each example as a singleton cluster at the outset and then successively merge pairs of clusters until all clusters have been merged into a single cluster that contains all examples. The predictive clustering trees and the agglomerative approaches produce binary tree hierarchies, while the balanced *k*-means clustering approach produces multi-branch tree hierarchies for $k > 2$.

In addition to these methods for hierarchy construction, we examine the generation of a random hierarchy. In particular, the labels are distributed evenly but randomly into the k subsets. The motivation here is to examine thoroughly the contribution of the hierarchy construction methods.

5. Predictive modelling for genre classification

We present the methodology used to construct predictive models for the task of genre classification using PCTs. We first present the general algorithm for constructing PCTs. Next, we otuline the specific PCTs able to predict all of the genres simultaneously but ignore the hierarchical information (i.e., address the task of genre prediction as a multi-label classification task). Furthermore, we give the PCTs able to predict all of the genres simultaneously and exploit the hierarchy information (i.e., address the task of genre prediction as a HMC task). We then discuss the PCTs that predict each genre separately: without considering the hierarchy (single-label classification - SC) or with considering the hierarchy (hierarchical single-label classification - HSC). Finally, we present the methods used to construct ensemble models: bagging and random forest.

5.1. Predictive clustering trees

The Predictive Clustering Trees (PCTs) framework views a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the CLUS system [23] – available for download at http://clus.sourceforge.net.

PCTs are induced with a standard *top-down induction of decision trees* (TDIDT) algorithm. The pseudo-code for the algorithm is outlined in Table 4. It takes as input a set of examples and outputs a tree. The heuristic that is used for selecting the tests is the reduction in variance caused by the partitioning of the instances corresponding to the tests. By maximizing the variance reduction, the cluster homogeneity is maximized and the predictive performance is improved. The main difference between the algorithm for learning PCTs and a standard decision tree learner is that the former considers the variance function and the prototype function (that computes a label for each leaf) as *parameters* that can be instantiated for a given learning task. PCTs have been instantiated for both MLC [23,31] and HMC [51].

PCTs for MLC. These can be considered as PCTs that are able to predict multiple binary (and thus discrete) targets simultaneously. Therefore, the variance function for the PCTs for MLC is computed as the sum of the Gini indices of the

Table 4
The top-down induction algorithm for PCTs.

procedure PCT	**procedure** BestTest
Input: A dataset E	**Input:** A dataset E
Output: A predictive clustering tree	**Output:** the best test (t^*), its heuristic score (h^*) and the partition ($\mathcal{P}^*$) it induces on the dataset (E)
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$	1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **if** $t^* \neq none$ **then**	2: **for each** possible test t **do**
3: **for each** $E_i \in \mathcal{P}^*$ **do**	3: $\mathcal{P} =$ partition induced by t on E
4: $tree_i = \text{PCT}(E_i)$	4: $h = Var(E) - \sum_{E_i \in \mathcal{P}} \frac{\|E_i\|}{\|E\|} Var(E_i)$
5: **return** $\text{node}(t^*, \bigcup_i \{tree_i\})$	5: **if** $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ **then**
6: **else**	6: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **return** $\text{leaf}(\text{Prototype}(E))$	7: **return** $(t^*, h^*, \mathcal{P}^*)$

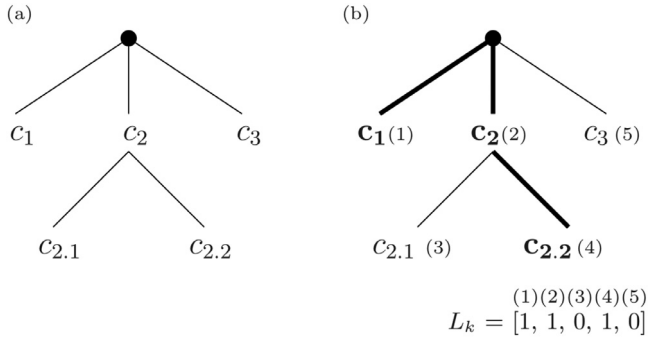

Fig. 4. Toy examples of a hierarchy structured as a tree. (a) Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2. (b) The set of classes $S_1 = \{c_1, c_2, c_{2.2}\}$, shown in bold, are represented as a vector (L_k).

target variables (T), i.e.,

$$Var(E) = \sum_{i=1}^{T} Gini(E, Y_i).$$

The prototype function returns a vector of probabilities that an instance belongs to a given class for each target variable. The most probable (majority) class value for each target can then be calculated by applying a threshold on these probabilities.

PCTs for HMC . The variance and prototype for PCTs for the HMC are defined as follows. First, the set of labels of each example is represented as a vector with binary components; the i'th component of the vector is 1 if the example belongs to class c_i and 0 otherwise. The variance of a set of examples E is defined as the average squared distance between each example's class vector (L_i) and the set's mean class vector ($\bar{L}$):

$$Var(E) = \frac{1}{|E|} \cdot \sum_{E_i \in E} d(L_i, \bar{L})^2.$$

The similarity at higher levels of the hierarchy is more important than the similarity at lower levels. Hence, the distance measure used is a weighted Euclidean distance:

$$d(L_1, L_2) = \sqrt{\sum_{l=1}^{|L|} w(c_l) \cdot (L_{1,l} - L_{2,l})^2},$$

where $L_{i,l}$ is the l^{th} component of the class vector L_i of an instance E_i, $|L|$ is the size of the class vector, and the class weights $w(c)$ decrease with the depth of the class in the hierarchy. More precisely, $w(c) = w_0 \cdot w(p(c))$, where $p(c)$ denotes the parent of class c and $0 < w_0 < 1$).

For example, consider the toy class hierarchy shown in Fig. 4(a,b), and two data examples: (X_1, S_1) and (X_2, S_2) that belong to the classes $S_1 = \{c_1, c_2, c_{2.2}\}$ (boldface in Fig. 4(b)) and $S_2 = \{c_2\}$, respectively. We use a vector representation with consecutive components representing membership in the classes c_1, c_2, $c_{2.1}$, $c_{2.2}$ and c_3, in that order (preorder traversal of the tree of class labels). The distance is then calculated as follows:

$$d(S_1, S_2) = d([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) = \sqrt{w_0 + w_0^2}.$$

The mean $\bar{L}$ of the class vectors of the examples in the leaf is stored as a prediction. Note that the value for the i^{th} component of $\bar{L}$ can be interpreted as the probability that an example arriving at the given leaf belongs to class c_i. The prediction for an example that arrives at the leaf can be obtained by applying a user defined threshold τ to the probability. Moreover, when a PCT makes a prediction, it preserves the hierarchy constraint (the predictions comply with the parent-child relationships from the hierarchy).

PCTs for SC. These PCTs are a special case of the PCTs for MLC where the size of the label space is 1. A separate local PCT (i.e., a regular decision tree) is learnt for each web genre, where web pages labeled with the given genre are considered as positive and all the others as negative examples.

PCTs for HSC. The HSC models take into account the hierarchical relationships among classes by virtue of training separate local classifiers only with the subset of examples which are labeled with a specific part of the class hierarchy [51]. More specifically, we construct a decision tree for each edge (connecting a class c with a parent class $par(c)$) in the hierarchy, thus creating an architecture of classifiers. The tree that predicts membership to class c is learnt using the instances that belong to $par(c)$. The resulting HSC tree architecture predicts the conditional probability $P(c|par(c))$. A new instance is predicted by recursive application of the product rule

$$P(c) = P(c|par(c)) \cdot P(par(c)),$$

starting from the tree for the top-level class. The obtained probabilities are thresholded to obtain the set of predicted classes similarly as for the PCTs for HMC to satisfy the hierarchy constraint.

5.2. Ensembles of PCTs

We consider ensembles of PCTs for structured prediction [23]. The PCTs in the ensembles are constructed by using the bagging [5] and random forests [6] methods that are often used in the context of decision trees. Bagging is an ensemble method that constructs the different classifiers by making bootstrap replicates of the training set and using each of these replicates to construct a predictive model. Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set, until an equal number of instances as in the training set is obtained.

A random forest is an ensemble of trees, where diversity among the predictors is obtained by using bootstrap replicates as in bagging, and additionally by changing the set of descriptive attributes during learning. More precisely, the PCT algorithm for tree construction uses a randomized version of the selection of attributes, which replaces the standard selection of attributes: At each node in the decision trees, a random subset of the descriptive attributes is taken, and the best attribute is selected from this subset. The number of attributes that are retained is given by a function f of the total number of descriptive attributes D (e.g., $f(D) = 1,\ f(D) = \lfloor\sqrt{D} + 1\rfloor,\ f(D) = \lfloor log_2(D) + 1\rfloor$ ...).

To construct ensemble models for the various machine learning tasks, corresponding type of PCTs are used as a base model. The prediction of an ensemble for a new instance is obtained by combining the predictions of all the base predictive models from the ensemble.

6. Experimental design

The comparison of the methods was performed using the CLUS system for predictive clustering. We constructed predictive models corresponding to the four types of modelling tasks: single-label classification (SLC), multi-label classification (MLC), hierarchical single-label classification (HSC), and hierarchical multi-label classification (HMC). For each modeling task, we constructed single tree models, and two ensemble models (bagging and random forest).

For the single tree models, we used *F*-test pruning to ensure that the produced models are not overfitted to the training data and have better predictive performance [51]. The exact Fisher test is used to check whether a given split/test in an internal node of the tree results in a statistically significant reduction in variance. If there is no such split/test, the node is converted to a leaf. A significance level is selected from the values 0.125, 0.1, 0.05, 0.01, 0.005 and 0.001 to optimize the predictive performance by using internal 3-fold cross validation. The parameters of the ensemble methods were instantiated following the recommendations from the literature. In particular, the number of models (classifiers) used in the ensembles is set to 100 and the voting scheme employed is probability distribution vote [2]. The size of the feature subsets needed for construction of the base classifiers for RF-PCT is set $f(x) = \lfloor 0.1 \cdot x + 1\rfloor$ [22]. The balanced k-means clustering method that is used for deriving the label hierarchies requires the number of clusters k. For this parameter, three different values (2, 3 and 4) were considered [30,49]. Furthermore, [49] have shown that balanced k-means outperforms random hierarchy construction for any value of k. Moreover, [48] have shown that data-based partitioning of the label space is always significantly better than random partitioning. Hence, we adopt the best value of k from the balanced k-means also for the random hierarchy construction.

The performance of the predictive models was evaluated using 3-fold cross-validation (as in the study that published the data [53]). We evaluate the predictive performance of the models on the leaf labels in the target hierarchy. In this way, we measure more precisely the influence of the inclusion of the hierarchies in the learning process on the predictive performance of the models.

We used 16 evaluation measures [31]: six *example-based* evaluation measures (*Hamming loss, accuracy, precision, recall,* F_1*score* and *subset accuracy*), six *label-based* evaluation measures (*micro precision, micro recall, micro* F_1, *macro precision, macro*

recall and *macro* F_1) and four *ranking-based* evaluation measures (*one-error, coverage, ranking loss* and *average precision*). These evaluation measures require predictions stating that a given label is present or not (binary 1/0 predictions). However, most predictive models predict a numerical value for each label and the label is predicted as present if that numerical value exceeds some pre-defined threshold. The performance of the predictive model thus directly depends on the selection of an appropriate value for the threshold. To this end, we applied a threshold calibration method by choosing the threshold that minimizes the difference in label cardinality between the training data and the predictions for the test data [38]. Note that, we do not use the output space of the test set while calculating the threshold.

The main focus of our experimental study is on elucidating novel knowledge about structuring the web genre space into hierarchical structure and its implications to the predictive performance of the predictive models exploiting that information. Note that knowledge plays a central role in the area of machine learning since its beginnings and machine learning is more than just performing "comparisons among the performance of algorithms that reveal little about the sources of power or the effects of domain characteristics" [26]. Next, there is a difference between practical and statistical significance [13] : "Statistical significance only implies that the outcome of a study is highly unlikely to have occurred as a result of chance, but it does no necessarily suggest that any difference or effect detected in a set of data is of any practical value." This means that sometimes differences that do not have real implications can be deemed statistically significant. All in all, we are more focused on the knowledge coming out of the study rather than the potential statistically significant differences in the performance.

7. Results and discussion

In this section, we present the results from the extensive experimental evaluation. The evaluation aims to answer several questions: (1) Does structuring of the output space improves the predictive performance? (2) Which data-driven hierarchy construction method yields hierarchy of web genres with best performance? (3) Is data-driven hierarchy construction better than expert hierarchy construction? (4) Which feature construction method yields the most discriminative features for web genre classification? (5) What is the influence of the ensemble models on the predictive power of structured and unstructured output space?

We discuss the results using 4 representative evaluation measures: *Hamming loss, micro* F_1, *macro* F_1 and *average precision*. The same conclusions will hold if all of the evaluation measures are used. Furthermore, the main discussion of the results covers the use of single PCT as a predictive model. We opted for this because in this way we can more precisely measure the influence of the different sources of information (both from structuring the output space and the different features). The ensemble models are used to emphasize the maximal potential of the output structures and feature sets (when using ensembles we could not make any conclusions as to where the predictive power comes from: the powerfull learning method or the information contained in the feature sets and the output structures). The complete results from the evaluation are available for download at the repository.

7.1. Structuring the output space

We examine two types of structure in the output space: multiple labels (i.e., a web page can belong to multiple web genres) and hierarchy on the labels (i.e., impose a structure on the web genres). We investigate in detail the influence of these two types of structure in the output space by considering four machine learning tasks that exploit differently the aforementioned structures in the output.

First, we discuss the influence of imposing a hierarchy on the output space. The results in Tables 5–8 clearly show that considering a hierarchy of web genres can improve the predictive performance. For the HSC task, these improvements are considerable and drastic. Next, the influence of the multiple labels is not that strong. When comparing the SLC and MLC tasks, notable performance improvement is made for the SANTINIS-ML dataset, while for the *20-genres* dataset this deteriorates the performance. We believe that this is the effect of the lower label cardinality of the *20-genres* dataset, as compared to the one of SANTINIS-ML: The multi-label effect would be more pronounced if there are datasets with annotations of multiple web genres per page. All in all, considering the web genre classification as a task of a HSC, i.e., imposing a hierarchy of the web genres, substantially improves the predictive performance.

7.2. Data-driven hierarchy construction

We use 4 methods for data-driven hierarchy construction: balanced k-means clustering (BkM), predictive clustering trees (PCTs), clustering with complete linkage (CL), and clustering with single linkage (SL). The quality of the constructed hierarchies needs to be assessed both for the HMC and HSC tasks - both of these exploit the hierarchy information.

We start by selecting the optimal value for the k parameter of the BkM method. The results reveal a limited influence of k to the predictive performance, especially for the HSC task. Nevertheless, we can note a slight advantage of setting k to 4, hence, we use that value for further analysis.

Second, we compare the performance obtained using the different hierarchies constructed with data-driven methods (the first four columns for the HMC and HSC tasks from Tables 5–8). The best performances per feature and task type are underlined (e.g., the best performance of *BOWSrf* features with the HMC task on the SANTINIS-ML dataset is obtained with

Table 5
The performance of the different machine learning tasks applied on the different features on the *20-genres* dataset measured with *Hamming loss* and *average precision*. The hierarchy construction method are abbreviated as follows: balanced *k*-means clustering (BkM), predictive clustering tree (PCT), clustering with complete linkage (CL), clustering with single linkage (SL), random (RND) and manual (MAN). The values for the *Hamming loss* mean less is better, while for *average precision* mean more is better. Both evaluation measures are in the interval [0,1].

HLoss	HMC						HSC						MLC	SLC
	CL	SL	BkM	PCT	RND	MAN	CL	SL	BkM	PCT	RND	MAN		
BOWSrf	0.105	0.11	0.084	*0.088*	*0.081*	0.088	0.013	0.013	0.013	0.014	0.013	0.012	0.092	*0.078*
TFIDFSrf	0.097	0.092	*0.082*	0.109	0.096	*0.086*	*0.011*	0.011	*0.01*	*0.012*	*0.01*	*0.01*	0.102	0.083
Struc	0.101	0.115	0.105	0.108	0.092	0.114	0.014	0.014	0.014	0.015	0.014	0.014	0.092	0.102
Pres	0.119	0.117	0.096	0.118	0.093	0.102	0.04	0.04	0.04	0.036	0.033	0.04	0.092	0.104
Context	0.097	*0.091*	0.092	0.091	0.094	0.1	0.074	0.074	0.065	0.065	0.063	0.065	*0.089*	0.087
PV_R	0.105	0.109	0.115	0.109	0.109	0.11	0.013	0.012	0.012	0.013	0.011	0.011	0.091	0.103
PV_C	0.125	0.114	0.103	0.095	0.095	0.105	*0.011*	*0.01*	*0.01*	*0.012*	*0.01*	*0.01*	0.091	0.097
PV_Gl_R	*0.093*	0.096	0.126	0.097	0.121	0.113	0.013	0.012	0.011	0.013	0.012	0.011	0.091	0.105
PV_Gl_C	0.11	0.109	0.12	0.106	0.136	0.093	0.012	0.011	0.011	*0.012*	0.011	0.011	0.091	0.102
Ngrams	0.089	0.105	0.091	0.095	0.092	0.091	0.014	0.014	0.013	0.014	0.014	0.013	0.091	0.086
average precision														
BOWSrf	*0.404*	0.402	0.453	*0.424*	0.451	0.446	0.986	0.986	0.986	0.984	0.987	0.986	0.284	*0.494*
TFIDFSrf	0.392	*0.41*	*0.478*	0.423	*0.464*	*0.459*	0.99	0.99	0.992	0.989	0.993	0.992	0.295	0.468
Struc	0.349	0.354	0.383	0.358	0.379	0.367	0.985	0.986	0.985	0.985	0.986	0.985	0.284	0.347
Pres	0.34	0.327	0.365	0.355	0.366	0.378	0.881	0.873	0.876	0.886	0.917	0.871	0.284	0.39
Context	0.344	0.33	0.387	0.364	0.398	0.407	0.648	0.65	0.648	0.656	0.648	0.648	*0.314*	0.388
PV_R	0.341	0.338	0.35	0.344	0.344	0.343	0.993	0.993	0.993	0.992	*0.994*	0.993	0.288	0.336
PV_C	0.347	0.335	0.363	0.355	0.377	0.367	*0.994*	*0.994*	*0.995*	*0.993*	0.993	*0.994*	0.288	0.387
PV_Gl_R	0.309	0.31	0.331	0.319	0.329	0.329	0.99	0.99	0.99	0.989	0.989	0.99	0.288	0.314
PV_Gl_C	0.35	0.351	0.361	0.353	0.352	0.359	0.992	0.993	0.994	0.992	0.991	0.993	0.288	0.344
Ngrams	0.393	0.398	0.42	0.379	0.43	0.413	0.982	0.982	0.983	0.981	0.982	0.983	0.288	0.471

Table 6
The performance of the different machine learning tasks applied on the different features on the *20-genres* dataset measured with *micro* F_1 and *macro* F_1. The hierarchy construction method are abbreviated as follows: balanced *k*-means clustering (BkM), predictive clustering tree (PCT), clustering with complete linkage (CL), clustering with single linkage (SL), random (RND) and manual (MAN). For both evaluation measures, smaller values mean better performance. Both evaluation measures are in the interval [0,1].

micro F_1	HMC						HSC						MLC	SLC
	CL	SL	BkM	PCT	RND	MAN	CL	SL	BkM	PCT	RND	MAN		
BOWSrf	*0.261*	0.264	0.303	*0.276*	0.306	0.297	0.9	0.905	0.905	0.894	0.904	0.909	0.086	*0.396*
TFIDFSrf	0.257	*0.279*	*0.327*	0.265	*0.32*	*0.314*	*0.919*	0.918	0.921	0.911	*0.929*	0.922	*0.125*	0.383
Struc	0.218	0.21	0.255	0.215	0.23	0.236	0.894	0.898	0.9	0.89	0.898	0.9	0.086	0.226
Pres	0.207	0.185	0.202	0.219	0.203	0.224	0.709	0.702	0.707	0.727	0.747	0.701	0.086	0.255
Context	0.201	0.18	0.248	0.22	0.262	0.265	0.48	0.481	0.475	0.486	0.479	0.48	0.118	0.215
PV_R	0.185	0.184	0.212	0.208	0.206	0.204	0.903	0.91	0.915	0.903	0.915	0.917	0.093	0.233
PV_C	0.213	0.187	0.214	0.2	0.215	0.217	0.917	*0.923*	*0.927*	*0.912*	0.924	*0.927*	0.093	0.278
PV_Gl_R	0.157	0.164	0.191	0.18	0.19	0.187	0.906	0.912	0.915	0.9	0.91	0.918	0.093	0.216
PV_Gl_C	0.203	0.215	0.223	0.218	0.223	0.177	0.912	0.917	0.922	0.91	0.916	0.922	0.093	0.24
Ngrams	0.218	0.266	0.273	0.222	0.281	0.245	0.897	0.899	0.901	0.893	0.897	0.902	0.093	0.362
macro F_1														
BOWSrf	0.177	0.176	0.202	0.17	0.209	0.206	0.897	0.905	0.904	0.895	0.904	0.909	0.018	0.415
TFIDFSrf	*0.189*	*0.203*	*0.239*	*0.187*	*0.271*	*0.237*	*0.918*	*0.918*	*0.92*	*0.914*	*0.926*	*0.921*	0.037	0.41
Struc	0.104	0.111	0.142	0.077	0.12	0.125	0.885	0.89	0.893	0.883	0.891	0.893	0.018	0.218
Pres	0.118	0.063	0.115	0.126	0.12	0.149	0.725	0.711	0.712	0.732	0.734	0.711	0.018	0.198
Context	0.101	0.051	0.174	0.118	0.193	0.207	0.493	0.495	0.484	0.493	0.486	0.488	*0.06*	0.199
PV_R	0.086	0.084	0.091	0.12	0.078	0.08	0.886	0.895	0.902	0.89	0.905	0.907	0.019	0.233
PV_C	0.134	0.117	0.125	0.104	0.134	0.166	0.906	0.914	*0.92*	0.907	0.919	0.92	0.019	0.285
PV_Gl_R	0.057	0.066	0.098	0.048	0.079	0.084	0.89	0.898	0.902	0.888	0.9	0.907	0.019	0.199
PV_Gl_C	0.093	0.111	0.099	0.104	0.094	0.065	0.9	0.908	0.913	0.903	0.909	0.914	0.019	0.234
Ngrams	0.116	0.156	0.209	0.112	0.22	0.177	0.902	0.904	0.906	0.899	0.905	0.907	0.019	0.39

a *CL* hierarchy, hence the top left value in Table 7 is underlined). For the *20-genres* dataset (Tables 5 and 6), we can note that BkM yelds the best performance on *micro* F_1, *macro* F_1 and *average precision* across the majority of the feature sets for both HMC and HSC. The *Hamming loss* measured on the HSC task coincides with the findings for the other measures, while for the HMC task does not provide clear recommendations. For the *SANTINIS-ML* dataset, there are some differences in the performance of the hierarchy construction methods on the HMC and HSC tasks. On one hand, the HSC task clearly

Table 7
The performance of the different machine learning tasks applied on the different features on the *SANTINIS-ML* dataset measured with *Hamming loss* and *average precision*. The hierarchy construction method are abbreviated as follows: balanced *k*-means clustering (BkM), predictive clustering tree (PCT), clustering with complete linkage (CL), clustering with single linkage (SL), random (RND) and manual (MAN). The values for the *Hamming loss* mean less is better, while for *average precision* mean more is better. Both evaluation measures are in the interval [0,1].

HLoss	HMC						HSC						MLC	SLC
	CL	SL	BkM	PCT	RND	MAN	CL	SL	BkM	PCT	RND	MAN		
BOWSrf	*0.1*	*0.101*	*0.103*	*0.101*	*0.103*	*0.103*	0.012	0.011	0.012	0.016	0.014	0.012	0.109	*0.109*
TFIDFSrf	*0.1*	0.105	*0.103*	*0.101*	0.105	0.103	*0.011*	*0.01*	0.011	*0.012*	*0.011*	*0.01*	0.109	0.114
Struc	0.104	0.106	0.104	0.109	0.105	0.104	0.012	0.011	0.012	0.013	0.011	*0.011*	0.114	0.119
Pres	0.103	0.118	0.107	0.103	0.107	0.107	0.047	0.048	0.047	0.044	0.052	0.048	0.144	0.11
Context	0.113	0.113	0.109	0.11	0.106	0.109	0.082	0.082	0.082	0.085	0.088	0.083	0.129	0.136
PV_R	0.119	0.119	0.11	0.11	0.108	0.116	*0.011*	*0.01*	*0.01*	0.013	0.012	0.011	*0.108*	0.126
PV_C	0.118	0.117	0.111	0.114	0.113	0.111	*0.011*	*0.01*	*0.01*	0.013	0.012	*0.01*	0.117	0.128
PV_Gl_R	0.126	0.117	0.122	0.125	0.118	0.122	*0.011*	*0.01*	*0.01*	0.013	0.012	*0.01*	0.136	0.13
PV_Gl_C	0.123	0.124	0.115	0.122	0.115	0.116	*0.011*	*0.01*	*0.01*	0.013	0.012	*0.01*	0.129	0.129
Ngrams	0.117	0.115	0.122	0.115	0.118	0.108	0.016	0.014	0.014	0.015	0.013	0.015	0.113	0.12
average precision														
BOWSrf	*0.716*	*0.718*	*0.716*	*0.697*	*0.713*	*0.715*	0.994	0.995	0.995	0.99	0.991	0.995	*0.697*	0.601
TFIDFSrf	0.715	0.705	0.709	*0.697*	0.71	0.707	0.994	0.994	0.992	0.994	0.994	0.994	*0.697*	0.59
Struc	0.705	0.7	0.696	0.681	0.693	0.696	0.994	0.995	0.994	0.994	0.996	0.995	0.689	0.57
Pres	0.679	0.689	0.682	0.678	0.682	0.684	0.929	0.927	0.932	0.946	0.914	0.926	0.66	0.631
Context	0.659	0.658	0.661	0.662	0.662	0.661	0.803	0.804	0.797	0.796	0.777	0.797	0.658	*0.651*
PV_R	0.658	0.658	0.676	0.656	0.667	0.676	0.996	*0.997*	*0.997*	0.995	*0.997*	*0.997*	0.662	0.544
PV_C	0.664	0.654	0.669	0.668	0.67	0.669	0.996	0.996	0.996	0.995	0.993	0.995	0.658	0.527
PV_Gl_R	0.641	0.642	0.658	0.653	0.653	0.658	0.995	0.996	0.996	*0.996*	0.995	0.996	0.661	0.525
PV_Gl_C	0.642	0.648	0.676	0.663	0.676	0.674	*0.997*	0.996	*0.997*	*0.996*	0.996	0.996	0.664	0.526
Ngrams	0.66	0.66	0.662	0.663	0.66	0.661	0.988	0.99	0.989	0.989	0.99	0.99	0.661	0.525

Table 8
The performance of the different machine learning tasks applied on the different features on the *SANTINIS-ML* dataset measured with *micro* F_1 and *macro* F_1. The hierarchy construction method are abbreviated as follows: balanced *k*-means clustering (BkM), predictive clustering tree (PCT), clustering with complete linkage (CL), clustering with single linkage (SL), random (RND) and manual (MAN). For both evaluation measures, smaller values mean better performance. Both evaluation measures are in the interval [0,1].

micro F_1	HMC						HSC						MLC	SLC
	CL	SL	BkM	PCT	RND	MAN	CL	SL	BkM	PCT	RND	MAN		
BOWSrf	*0.545*	*0.565*	0.546	0.524	*0.545*	0.544	0.944	0.949	0.946	0.929	0.938	0.948	*0.54*	*0.517*
TFIDFSrf	0.543	0.558	*0.547*	*0.526*	*0.545*	*0.545*	*0.953*	0.953	0.949	*0.946*	*0.951*	*0.954*	*0.54*	0.497
Struc	0.52	0.516	0.522	0.508	0.528	0.521	0.948	0.95	0.948	0.943	0.949	0.949	0.503	0.473
Pres	0.466	0.512	0.494	0.46	0.494	0.495	0.795	0.785	0.795	0.803	0.765	0.785	0.514	0.449
Context	0.468	0.467	0.446	0.44	0.431	0.446	0.637	0.642	0.639	0.623	0.609	0.631	0.456	0.474
PV_R	0.484	0.485	0.493	0.507	0.512	0.497	0.952	0.955	0.953	0.944	0.948	0.951	0.467	0.444
PV_C	0.473	0.484	0.478	0.507	0.487	0.481	0.951	0.955	*0.956*	0.942	0.948	*0.954*	0.49	0.429
PV_Gl_R	0.464	0.454	0.492	0.484	0.478	0.492	0.952	*0.956*	0.954	0.944	0.948	*0.954*	0.465	0.424
PV_Gl_C	0.464	0.473	0.503	0.5	0.503	0.504	0.952	0.955	0.954	0.942	0.946	*0.954*	0.471	0.442
Ngrams	0.463	0.463	0.507	*0.526*	0.512	0.459	0.93	0.938	0.936	0.932	0.942	0.934	0.459	0.465
macro F_1														
BOWSrf	*0.165*	0.171	0.165	0.165	0.189	0.178	0.851	*0.854*	*0.852*	0.841	0.821	*0.852*	*0.158*	*0.266*
TFIDFSrf	*0.165*	*0.195*	*0.182*	0.164	*0.197*	*0.18*	0.847	0.848	0.845	0.84	0.845	0.848	*0.158*	0.257
Struc	0.148	0.156	0.148	*0.171*	0.154	0.147	*0.852*	0.853	0.851	0.838	0.852	*0.852*	0.129	0.205
Pres	0.137	0.142	0.142	0.099	0.142	0.142	0.577	0.568	0.574	0.573	0.541	0.557	0.148	0.14
Context	0.081	0.081	0.077	0.073	0.059	0.077	0.377	0.379	0.374	0.398	0.385	0.367	0.079	0.111
PV_R	0.153	0.15	0.142	0.149	0.156	0.156	0.846	0.848	0.845	0.829	0.777	0.842	0.102	0.194
PV_C	0.126	0.136	0.134	0.133	0.137	0.144	0.803	0.806	0.806	0.789	0.787	0.805	0.13	0.189
PV_Gl_R	0.121	0.118	0.132	0.131	0.124	0.132	0.839	0.84	0.836	0.817	0.782	0.836	0.082	0.18
PV_Gl_C	0.131	0.127	0.146	0.141	0.146	0.152	0.792	0.794	0.794	0.78	0.772	0.791	0.109	0.198
Ngrams	0.138	0.127	0.166	0.151	0.155	0.134	0.847	0.851	0.851	*0.844*	*0.854*	0.85	0.098	0.226

prefers the clustering with single linkage (SL) over the other methods across all evaluation measures. On the other hand, in the HMC task, the hierarchy construction methods perform differently when coupled with the different feature sets and evaluated on the different measures. Nonetheless, SL and BkM methods stand out as the best ones. All in all, the balanced *k*-means method for hierarchy construction offers the best performance across datasets, feature sets and tasks.

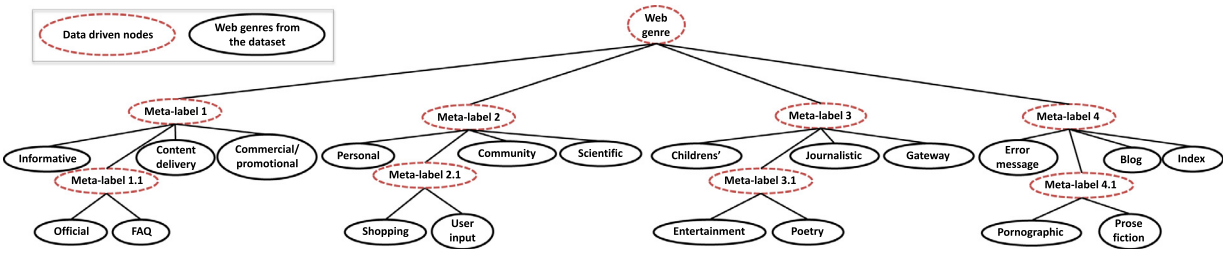

Fig. 5. Web genre hierarchy constructed by the balanced k-means algorithm (for k=4) for the *20-Genre* dataset.

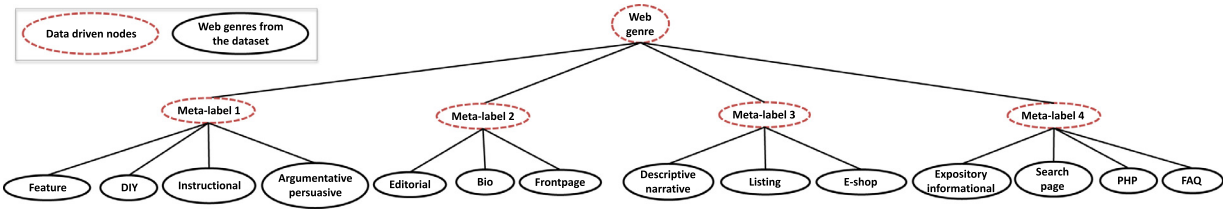

Fig. 6. Web genre hierarchy constructed by the balanced k-means algorithm (for k=4) for the *SANTINIS-ML* dataset.

Furthermore, we compare the performance of a random generated hierarchy of web genres to the performance of the hierarchy obtained with BkM. The results reveal that for the HMC task there are no differences in performance between the hierarchies obtained with these two methods on both datasets. Similar conclusions can be made for the HSC task on the *20-genres* dataset, while the SANTINIS-ML dataset reveals some advantage of using BkM to construct the hierarchy. The practical implications of this important finding are that a randomly generated hierarchy can produce a very good predictive performance.

7.3. Data- vs expert-driven hierarchy

We next investigate the performance of the data-driven and the expert-driven hierarchy considering the four evaluation measures. More specifically, we compare the performance of the BkM hierarchy with the performance of the hierarchy derived by an expert. For the *20-genres* dataset, the HMC and HSC prefer different hierarchies: HMC prefers the BkM hierarchy, while HSC prefers the expert-constructed hierarchy. For the SANTINIS-ML dataset, the HSC prefers the BkM hierarchy, while the results for the HMC task are somewhat inconclusive.

We next examine the different hierarchies. Namely, we visually inspect the expert-based hierarchies and juxtapose them with the ones obtained with balanced *k*-means (and *k* set to 4). The expert and data-driven hierarchies for the *20-genres* datasets are depicted in Figs. 2 and 5, respectively. We can note that these two hierarchies of web genres differ from each other: the semantic similarities captured by the expert in the former hierarchy are not present in the latter hierarchy. Figs. 3 and 6 depict the expert and data-driven hierarchies, respectively, for the SANTINIS-ML dataset. Although the hierarchies are somewhat different to each other, here, some of the semantic similarities outlined by the expert are preserved by the data-driven hierarchy construction method. For example, the *Editorial* and *Frontpage* web genres appear together in the both hierarchies, as well as the *DIY* and *Instructional* web genres.

To summarize, constructing a hierarchy of web genres substantially improves the predictive performance of web genre classification. In other words, considering the web genre classification as a hierarchical task yields performance improvement. If for a given dataset an expert-constructed hierarchy is provided, the users could use that hierarchy and obtain competitive predictive performance. If there is no such hierarchy provided one can then use balanced *k*-means to construct a hierarchy. However, the simplest and fastest course of action would be to generate a random hierarchy and exploit it to obtain better predictive performance.

7.4. Feature sets for web genres

The various feature extraction techniques are designed to capture different aspects of the web pages. These features are then used in a context of the different prediction tasks (SC, MLC, HSC or HMC), which produces different datasets

and models with different predictive performance. We discuss the results from the aspect of the different tasks. The best performance of a given output structure across all of the feature sets is marked in italics in Tables 5–8. In this way, we elucidate the best features by looking at the ones with the most italics in the respective rows.

To begin with, for the SLC task, the best performing features are the *BOWSrf* features for the both datasets. Second, for the MLC task, on the *20-genres* dataset the *Context* features yield the best result, while on the SANTINIS-ML dataset best features are *BOWSrf* and *TFIDFSrf* (i.e., the surface features). Next, the HSC task produces best results when it is used jointly with *TFIDTSrf* or *PV_C* features on both datasets. Finally, the HMC task prefers coupling with the surface type features (*BOWSrf* and *TFIDTSrf*) for producing the best results.

The results reveal that the simpler tasks (SC and MLC) rely on features that capture some simple properties of the webpages (*Context* features consider the page URL and the present hyperlinks), while the more complex tasks are able to fully exploit the more complex ('black box' style) feature types. Furthermore, we examine the influence of removing the HTML tags for the PV features. For the *20-genres* dataset, the results show some competitive advantage of using clean over raw web pages, while for the SANTINIS-ML dataset there is no difference in performance. All in all, the recommended course of action is that the use of surface and/or paragraph vector type of features will yield very good results.

7.5. Ensemble models for web genre classification

We constructed two types of ensemble models using bagging and random forests. Both ensembles achieve very good predictive performance, especially the ensemble models on the HSC task. We discuss the results in more detail, while complete tables with results are given in the repository.

To begin with, both ensembles are considerably better than single tree models across the two datasets, the four tasks and the 16 evaluation measures. Moreover, there is no notable difference between the performance of the two ensembles: on some cases bagging is better, while on other random forests are better.

Second, considering multiple web genres per page does not change the predictive performance in the tasks without hierarchy of genres (i.e., comparing MLC vs SC). Contrary to this, considering both the hierarchy and multiple annotations jointly (HMC) does not improve the performance over the tasks that do not consider the web genres hierarchy. The best overall performance (with a remarkable margin) is obtained when the hierarchy is considered but without the multiple web genres annotations (the HSC task).

Next, the type of hierarchy considered does not influence strongly the predictive performance. In general, both ensemble methods on the *20-genres* dataset yield better results when the random hierarchy is used in HMC, while for HSC the type of hierarchy is not influential. For the SANTINIS-ML dataset, the results reveal that both ensemble methods prefer the hierarchy obtained with single linkage (SL) for the HSC task, while for the HMC task the type of hierarchy is not that important.

Finally, the different feature types influence the performance of both the ensemble models. On the *20-genres* dataset, the SC, MLC and HMC tasks yield best performance when coupled with *TFIDFSrf* features, while HSC yields best performance when coupled with anyone of the paragraph vector features. On the SANTINIS-ML dataset, the SC, MLC and HMC tasks prefer the surface features (*BOWSrf* and *TFIDTSrf*), while HSC prefers the paragraph vector features with pre-trained model, especially the *PV_Gl_R* features. Finally, the comparison of clean over raw web pages for the PV features reveals that there is no clear difference in the overall performance.

8. Conclusions

In this paper, we advocate a new approach for addressing the task of web genres classification. Traditionally, this task is treated as a multi-class problem, while there are few recent studies that advise to treat it as a structured output prediction problem, either by considering multiple web genres per web page or exploiting a hierarchy of web genres. We investigate the possibility of structuring the output space in detail. First of all, we translate the task of web genre prediction into four different machine learning tasks – each exploiting different levels of structure in the output space. Namely, we consider the following tasks: single-label classification (SLC, each web page is annotated with single genre), multi-label classification (MLC, each web page is annotated with multiple genres), hierarchical single-label classification (HSC, each page is annotated with a single genre and the genres are organized into a hierarchy) and hierarchical multi-label classification (HMC, each page is annotated with multiple genres organized into a genre hierarchy).

Next, we consider four data-driven methods for hierarchy construction, which are based on web genre co-occurrences in the web pages. More specifically, we investigated balance k-means (with three values for k), predictive clustering trees, agglomerative clustering with single and complete linkage. Moreover, we constructed a random hierarchy to measure the influence of the hierarchy construction methods. Finally, we also proposed two hierarchies constructed by an expert.

Furthermore, we use several feature extraction techniques for representing web pages. Namely, we use 5 types of features based on linguistic properties of the web pages, 4 feature types using neural network models and 1 feature type based on character n-grams.

For proper assessment of the different influences, we use the predictive modelling methodology that is able to grasp the different tasks described above with a unique modelling paradigm – predictive clustering trees. Single PCT models were used for correct measuring the influence of the different hierarchies and feature sets, while two types of ensemble models (obtained with bagging and random forests) were used to obtain state-of-the-art predictive performance.

The experimental evaluation is performed on two benchmark corpora for web genre prediction: *20-genre* and SANTINIS-ML. These corpora are the only ones available with multiple annotations per web page. The results of the evaluation can be summarized by answering the experimental questions:

1. *Does structuring of the output space improves the predictive performance?*
   Introducing structure in the output space yields models with better predictive performance. HSC is the best performing task across both datasets, all predictive models, all feature sets and all different hierarchies.
2. *Which data-driven hierarchy construction method yields hierarchy of web genres with best performance?*
   The balanced *k*-means method for hierarchy construction offers the best performance across datasets, feature sets and tasks. Single linkage clustering (SL) coupled with HSC also yields very good predictive performance.
3. *Is data-driven hierarchy construction better than expert hierarchy construction?*
   Data-driven hierarchy construction is at least as good as expert-constructed hierarchy. Moreover, random hierarchy achieves very good predictive performance (hence, if the goal is predictive power, there is no need for experts' efforts for construction of the 'perfect' hierarchy).
4. *Which feature construction method yields the most discriminative features for web genre classification?*
   Surface and paragraph vector features are the best feature sets – they offer the best predictive performance.
5. *What is the influence of the ensemble models on the predictive power of structured and unstructured output space?*
   Both ensemble models offer state-of-the-art predictive performance and they have a superior predictive performance than single tree models. Bagging and random forests perform equally well. The best performance is obtained when using ensembles on the HSC task coupled with paragraph vector features.

In summary, from an user perspective, when a new web genres corpora becomes available, one needs to describe the web pages using paragraph vector features (without wasting efforts to remove the HTML tags, since it does not influence the final result), construct a random hierarchy of web genres (this by-passes the need for experts' effort) and approach the task as hierarchical single-label classification.

An important challenge that should be addressed to make a web genre classifier a widely accepted addition to search engines is: how to construct a classifier that covers a multitude of genres present on the internet. We consider our approach for automatic construction of a web genre hierarchy as a step forward in addressing this challenge. It enables construction of hierarchy over both well-established and novel genres without an effort of a genre expert. Moreover, CLUS system based on PCTs is a versatile machine learning system that incorporates PCTs for single-label, multi-label and hierarchical multi-label classification, as well as semi-supervised learning and unsupervised clustering. We plan to use it in future experiments to dynamically adapt web genre classifier with unlabeled web pages and to detect genres absent from an initial genre palette.

We plan to extend this work in several directions. First, we plan to develop hierarchies of web genres structured as directed acyclic graphs, which seems more natural in modelling relations between genres. It could also be useful to design a hierarchy construction algorithm to break down existing genres into sub-genres. For example, 'Journalistic' is heterogeneous genre that is composed of subgenres news, reportage, editorial, interview and review and the classifier may benefit from breaking such genre categories into smaller sub-categories. Furthermore, we will examine the complementarity of the different feature sets and explore potential feature set combinations to further improve the predictive performance. Finally, we will investigate the random generation of hierarchies and its influence on the predictive performance in terms of stability of the results and the influence of the topology of the hierarchy on the performance.

Conflict of interest

We wish to confirm that there are no conflicts of interest associated with this publication and there has been no financial support for this work that could have influenced its outcome.

Acknowledgments

We acknowledge the financial support of the European Commission through the grant ICT-2013-612944 MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data; from the Republic of Slovenia and the European Union under the European Regional Development Fund (grant Raziskovalci-2.0-FIŠ-529008, implementation of the operation no. C3330-17-529008) and from the Slovenian Research Agency grants P2-0103 and J2-9230.

Appendix

Fig. 7. The web page from the 20-genre collection that expresses a personal opinion of blog's author, sharing conventions of both 'Blog' and 'Personal' genres. It should be noted that this is the top part of the web page with the majority of the web page content. The complete page is in the 20-genre collection in file0797.htm.

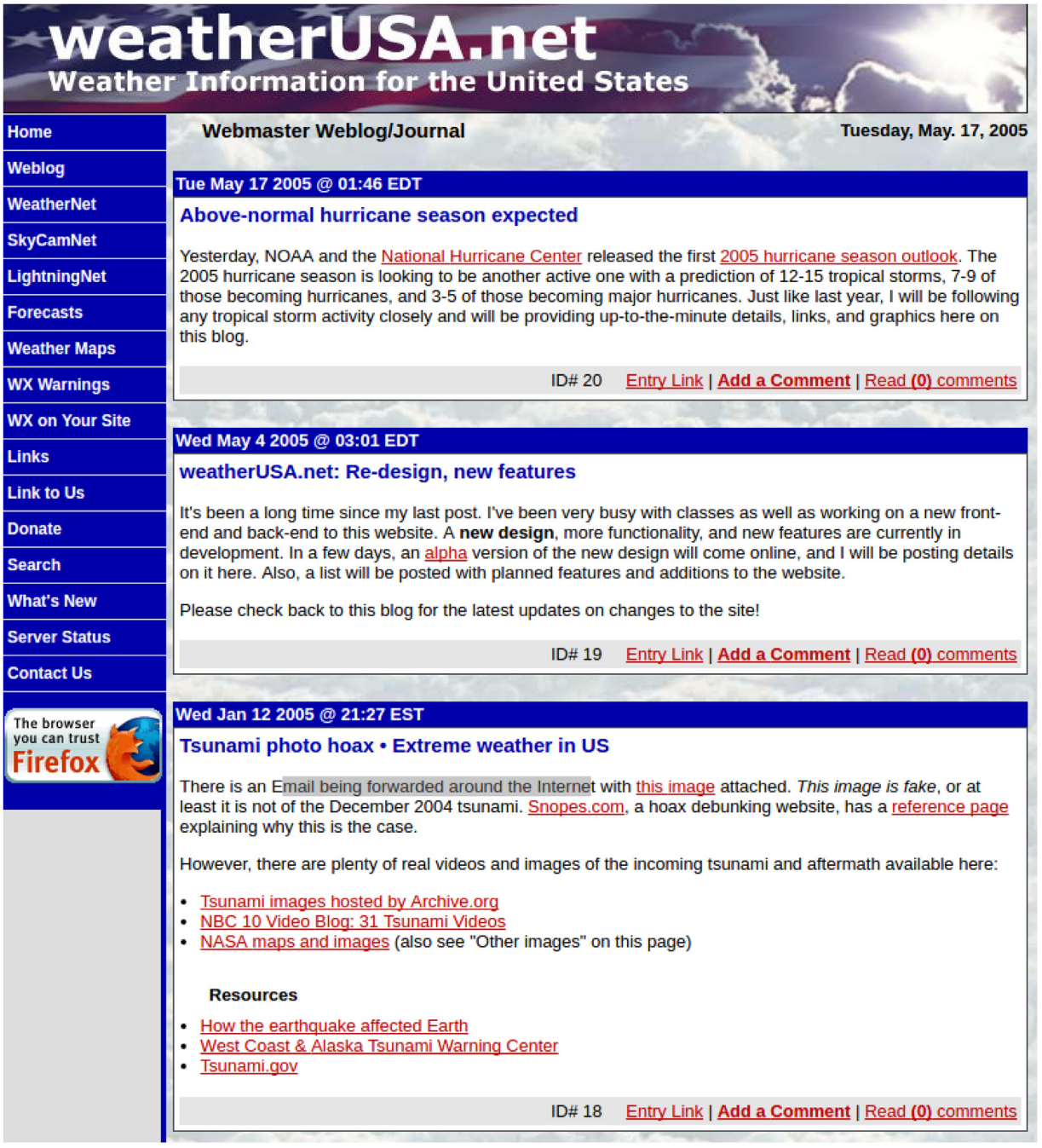

Fig. 8. The web page from the 20-genre collection that contains news on weather conditions and belongs to both 'Blog' and 'Journalistic' genres. It should be noted that this is the topmost part of the web page. The complete page is in the 20-genre collection in file0235.htm.

Simon Willison's Weblog

PHP, Python, CSS, XML and general web development

Skip to Navigation

HTMLifying user input

I've added a comment system to my new Kansas blog. Since the target audience for that site is friends and family rather than fellow web developers, I've taken a very different approach to processing the input from comments. While this blog insists upon valid XHTML and gives very little help to comment posters aside from highlighting validation problems, my new site's comment system takes the more traditional root of disallowing HTML while automatically converting line breaks and links.

The standard way of doing this with PHP is to use the `nl2br` function. I've never been a big fan of this method as I prefer blocks of text to be surrounded by paragraph tags. Luckily, adding paragraph tags to blocks of text is a relatively easy task. Here's the pseudo-code, mocked up in Python because it's quicker to experiment with than PHP:

```
>>> text = '''... lengthy text block here ...'''
>>> paras = text.split('\n\n')
>>> paras = ['<p>%s</p>' % para.strip() for para in paras]
>>> print '\n\n'.join(paras)
```

The above code splits the text block on any occurrence of a double newline, then wraps each of the resulting blocks in a paragraph tag (after stripping off any remaining whitespace) before joining the blocks back together with a pair of newlines between each one - because I like to keep my HTML nicely formatted. What it doesn't do is handle any necessary
 tags. The trick now is to replace any single line breaks with
 without interfering with the paragraph tags. The easiest way to do this is to put the replacement inside the loop, so that only line breaks that occur within a paragraph are replaced. Here's the updated list comprehension:

```
>>> paras = ['<p>%s</p>' % p.strip().replace('\n', '<br>\n') for p in paras]
```

The final job is to convert the above in to PHP:

```
$paras = explode("\n\n", $text);
for ($i = 0, $j = count($paras); $i < $j; $i++) {
    $paras[$i] = '<p>'.
        str_replace("\n", "<br>\n", trim($paras[$i])).
        '</p>';
}
$text = implode("\n\n", $paras);
```

That's the line conversions handled, but there are a few other important steps. Any HTML tags entered by the user need to be either stripped out or disabled by converting them to entities. Converting them to entities carries the risk of ugly failed attempts at HTML appearing on the comments page, but stripping tags carries an equal risk of innocent parts of a legitimate comment (such as a <wink>) being discarded. I chose to go the entity conversion route but force commenters to preview their comments before posting them, a trick I picked up from Adrian's blog. The final step is to automatically convert links in to `<a href="">` tags. I achieve this using a pair of naive regular expressions in the hope that the preview screen would avoid them mangling comments in a way not intended by the author.

Here's the finished PHP function:

```
function untrustedTextToHTML($text) {
    $text = htmlentities($text);
    $paras = explode("\n\n", $text);
    for ($i = 0, $j = count($paras); $i < $j; $i++) {
        $paras[$i] = '<p>'.
            str_replace("\n", "<br>\n", trim($paras[$i])).
            '</p>';
    }
    $text = implode("\n\n", $paras);
    // Convert http:// links
    $text = preg_replace('|\\b(http://[^\s)<]+)|',
        '<a href="$1">$1</a>', $text);
    // Convert www. links
    $text = preg_replace('|\\b(www.[^\s)<]+)|',
        '<a href="http://$1">$1</a>', $text);
    return $text;
}
```

I have no doubt it could be improved, but my tests so far have shown it to be good enough for the job at hand.

Posted 19th October 2003 - 02:53 | Categories: PHP, Python

Fig. 9. The web page from the 20-genre collection that represents a blog with a computer programming how-to. The page belongs to both 'Blog' and 'Informative' genres. It should be noted that this is the topmost part of the web page. The complete page is in the 20-genre collection in file1112.htm.

References

[1] N.R. Asheghi, S. Sharoff, K. Markert, Crowdsourcing for web genre annotation, Lang. Resour. Eval. 50 (3) (2016) 603–641.
[2] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Mach. Learn. 36 (1) (1999) 105–139.
[3] V. Berninger, Y. Kim, S. Ross, Building a Document Genre Corpus: A Profile of the krys i Corpus, Workshop Held in Conjunction with IliX 2008, 2008.
[4] H. Blockeel, L.D. Raedt, J. Ramon, Top-Down Induction of Clustering Trees, in: Proc. of the 15th International Conference on Machine Learning, 1998, pp. 55–63.
[5] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[6] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[7] M. Breskvar, D. Kocev, S. Džeroski, Multi-Label Classification using Random Label Subset Selections, in: Discovery Science, Springer International Publishing, 2017, pp. 108–115.
[8] G. Chen, B. Choi, Web Page Genre Classification, in: Proceedings of the 2008 ACM Symposium on Applied Computing, in: SAC '08, ACM, 2008, pp. 2353–2357.
[9] K. Crowston, B. Kwaśnik, J. Rubleske, Problems in the Use-Centered Development of a Taxonomy of Web Genres, in: Genres on the Web, Springer, 2010, pp. 69–84.
[10] H. Cunningham, V. Tablan, A. Roberts, K. Bontcheva, Getting more out of biomedical documents with gate's full lifecycle open source text analytics, PLoS Comput. Biol. 9 (2) (2013) e1002854.
[11] J.F. Da Silva, G. Dias, S. Guilloré, J.G.P. Lopes, Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units, in: Progress in Artificial Intelligence, Springer, 1999, pp. 113–132.
[12] I. Dimitrovski, D. Kocev, S. Loskovska, S. Džeroski, Improving bag-of-visual-words image retrieval with predictive clustering trees, Inf. Sci. 329 (2016) 851–865.
[13] S. Garca, A. Fernndez, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.
[14] M. Hepple, Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-Based pos Taggers, in: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2000. 278–277
[15] A.S. Hornby, S. Wehmeier, M. Ashby, Oxford Advanced Learner's Dictionary, 1428, Oxford University Press Oxford, 1995.
[16] R.S. J. Pennington, C. Manning, Glove: Global Vectors for Word Representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[17] C. Jebari, MLICC: A Multi-Label and Incremental Centroid-Based Classification of Web Pages by Genre, in: Natural Language Processing and Information Systems, Springer, 2012, pp. 183–190.
[18] H. Joho, M. Sanderson, The SPIRIT Collection: An Overview of a Large Web Collection, in: ACM SIGIR Forum, 38, ACM, 2004, pp. 57–61.
[19] A. Joly, P. Geurts, L. Wehenkel, Random Forests with Random Projections of the Output Space for High Dimensional Multi-Label Classification, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2014, pp. 607–622.
[20] I. Kanaris, E. Stamatatos, Learning to recognize webpage genres, Inf. Process. Manag. 45 (5) (2009) 499–512.
[21] J. Karlgren, D. Cutting, Recognizing Text Genres with Simple Metrics using Discriminant Analysis, in: Proceedings of the 15th conference on Computational linguistics-Volume 2, Association for Computational Linguistics, 1994, pp. 1071–1075.
[22] D. Kocev, Ensembles for Predicting Structured Outputs PhD thesis,, Jožef Stefan International Postgraduate School, 2011. Ljubljana, Slovenia.
[23] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Tree ensembles for predicting structured outputs, Pattern Recognit. 46 (3) (2013) 817–833.
[24] H. Kučera, W.N. Francis, Computational Analysis of Present-Day American English, Dartmouth Publishing Group, 1967.
[25] K.P. Kumari, A.V. Reddy, Performance improvement of web page genre classification, Int. J. Comput. Appl. 53 (10) (2012) 24–27.
[26] P. Langley, The changing science of machine learning, Mach. Learn. 82 (3) (2011) 275–279.
[27] Q.V. Le, T. Mikolov, Distributed representations of sentences and documents, (2014) arXiv:1405.4053.
[28] D. Lee, Genres, registers, text types, domains and styles: clarifying the concepts and navigating a path through the BNC jungle, Lang. Comput. 42 (1) (2002) 247–292.
[29] J. Levatić, D. Kocev, S. Džeroski, The importance of the label hierarchy in hierarchical multi-label classification, J. Intell. Inf. Syst. 45 (2) (2015) 247–271.
[30] G. Madjarov, I. Dimitrovski, D. Gjorgjevikj, S. Džeroski, The use of data-derived label hierarchies in multi-label classification, J. Intell. Inf. Syst. 47 (1) (2016) 57–90.
[31] G. Madjarov, D. Kocev, D. Gjorgjevikj, S. Džeroski, An extensive experimental comparison of methods for multi-label learning, Pattern Recognit. 45 (9) (2012) 3084–3104.
[32] G. Madjarov, V. Vidulin, I. Dimitrovski, D. Kocev, Web Genre Classification via Hierarchical Multi-Label Classification, in: Intelligent Data Engineering and Automated Learning–IDEAL 2015, Springer, 2015, pp. 9–17.
[33] R. Malhotra, A. Sharma, Quantitative evaluation of web metrics for automatic genre classification of web pages, Int. J. Syst. Assurance Eng. Manag. 8 (2) (2017) 1567–1579.
[34] C.D. Manning, P. Raghavan, H. Schütze, An Introduction to Information Retrieval, Cambridge University Press, 2009.
[35] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, (2013) arXiv:1301.3781.
[36] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed Representations of Words and Phrases and Their Compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.
[37] M.F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[38] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier Chains for Multi-Label Classification, in: Proc. of the 20th European Conference on Machine Learning, 2009, pp. 254–269.
[39] G. Rehm, M. Santini, A. Mehler, P. Braslavski, R. Gleim, A. Stubbe, S. Symonenko, M. Tavosanis, V. Vidulin, Towards a Reference Corpus of Web Genres for the Evaluation of Genre Identification Systems, LREC, 2008.
[40] M.A. Rosso, Using Genre to Improve Web Search Ph.D. thesis, University of North Carolina at Chapel Hill, 2005.
[41] M. Santini, Automatic Identification of Genre in Web Pages Ph.D. thesis, University of Brighton, 2007.
[42] M. Santini, Characterizing Genres of Web Pages: Genre Hybridism and Individualization, in: HICSS 2007: 40th Annual Hawaii International Conference on System Sciences, 2007. 71–71
[43] M. Santini, Zero, single, or multi? genre of web pages through the users perspective, Inf. Process. Manag. 44 (2) (2008) 702–737.
[44] M. Santini, Cross-Testing a Genre Classification Model for the Web, in: Genres on the Web, Springer, 2010, pp. 87–128.
[45] C.N. Silla Jr, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Mini. Know. Discov. 22 (1–2) (2011) 31–72.
[46] B. Stein, S.M. Zu Eissen, N. Lipka, Web Genre Analysis: Use Cases, Retrieval Models, and Implementation Issues, in: Genres on the Web, Springer, 2010, pp. 167–189.
[47] A. Stubbe, C. Ringlstetter, K.U. Schulz, Genre as noise: noise in genre, Int. J. Doc. Anal. Recognit. 10 (3–4) (2007) 199–209.
[48] P. Szymański, T. Kajdanowicz, K. Kersting, How is a data-driven approach better than random choice in label space division for multi-label classification? Entropy 18 (8) (2016) 282.
[49] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective and Efficient Multilabel Classification in Domains with Large Number of Labels, in: Proc. of the ECML/PKDD Workshop on Mining Multidimensional Data, 2008, pp. 30–44.
[50] G. Tsoumakas, I. Vlahavas, Random k-Labelsets: An Ensemble Method for Multilabel Classification, in: Proc. of the 18th European conference on Machine Learning, 2007, pp. 406–417.
[51] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2) (2008) 185–214.

[52] V. Vidulin, M. Luštrek, M. Gams, Using Genres to Improve Search Engines, in: Proc. of the Int. Workshop Towards Genre-Enabled Search Engines, 2007, pp. 45–51.
[53] V. Vidulin, M. Luštrek, M. Gams, Multi-label approaches to web genre identification, J. Lang. Tech. Comp. Linguist. 24 (1) (2009) 97–114.
[54] Z. Wu, K. Markert, S. Sharoff, Fine-Grained Genre Classification using Structural Learning Algorithms, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 749–759.