LIBER QUARTERLY
The Journal of the Association of European Research Libraries
Vol. 29, (2019) xx–xx | e-ISSN: 2213-056X

Annif: DIY Automated Subject Indexing Using Multiple Algorithms

Osma Suominen

National Library of Finland
osma.suominen@helsinki.fi, orcid.org/0000-0003-0042-0745

Abstract

Manually indexing documents for subject-based access is a labour-intensive process. We propose using metadata gathered from bibliographic databases to train algorithms that assist librarians in that work. We have developed Annif, an open source tool and microservice for automated subject indexing. After training it with a subject vocabulary and existing metadata, Annif can be used to assign subject headings for new documents. We have tested Annif with different document collections including scientific papers, old scanned books and contemporary e-books, Q&A pairs from an "ask a librarian" service, Finnish Wikipedia, and the archives of a local newspaper. The results of analysing scientific papers and current books have been reassuring, while other types of documents have proved to be more challenging. The current version is based on a combination of existing natural language processing and machine learning tools. By combining multiple approaches and existing open source algorithms, Annif can build on the strengths of individual algorithms and adapt to different settings. With Annif, we expect to improve subject indexing and classification processes especially for electronic documents as well as collections that otherwise would not be indexed at all.

Keywords: metadata; automated subject indexing; natural language processing; machine learning

Uopen Journals | http://liberquarterly.eu/ | DOI: 10.18352/lq.10285

1. Introduction

Libraries manage a vast amount of metadata about different kinds of documents. Typically, these documents are indexed with subject headings from a subject vocabulary such as a thesaurus or subject heading system to improve discoverability. Manually indexing documents is a very labour-intensive intellectual process. Many new documents are available electronically, so it is possible to have a machine perform part of the indexing based on either the full text or shorter pieces of text such as a summary, an abstract, or a descriptive title.

For the machine to perform well, it needs to be trained with examples. Libraries have a lot of training data in the form of bibliographic databases, but in many cases, only a title and possibly an abstract is available, but not the full text. We propose to leverage that data to help indexing new documents.

To do so, we have developed Annif, an open source multi-algorithm automated indexing tool. After loading a subject vocabulary and existing metadata, Annif learns how to assign subject headings to new documents. It can also be used as a web service that can be integrated with other systems. Annif is being developed on GitHub[1] and thanks to the collaboration between the Zenodo repository and GitHub, it also has a permanent DOI.[2] An initial prototype was developed in early 2017 and a new version that is suitable for production use is now ready to be used.

2. Background and Related Work

In this section we explain the typical process of operation of automated indexing systems, review the main approaches used in automated indexing and discuss how automated indexing services can be provided as web services that can be integrated with other systems. We will also consider the limitations of existing systems from the perspective of libraries.

2.1. Process of Automated Indexing

Automated subject indexing systems generally follow a particular process. First, text documents are preprocessed, for example by tokenizing the text into sentences and individual words, converting words into lower case, removing stop words and/or stemming or lemmatizing words so that different grammatical variations of the same word are reduced to the stem or lemma that identifies the meaning of the word. Second, the documents are often converted into a vector representation of word frequencies, known as a *bag-of-words model,* that can be used to query for matching subjects using a suitable algorithm. Alternatively, the preprocessed tokens may be directly matched with terms from a controlled vocabulary. In both cases, the result is a list of *candidate subjects* for the document. In order to determine the final set of suggested subjects for the document, the candidates must then be ranked and only the most promising ones retained (Medelyan, 2009; Toepfer & Seifert, 2018).

2.2. Approaches

Algorithms for automated subject indexing can generally be divided into *lexical* and *associative approaches* (Toepfer & Seifert, 2018). In lexical approaches, frequently occurring or otherwise salient terms in the document are matched with terms in the vocabulary. Such algorithms can be relatively simple and precise, but their downside is that quite often not all relevant subjects appear verbatim in document text, so these will never be suggested by lexical algorithms (Pouliquen, Steinberger, & Ignat, 2003). Associative approaches, including machine learning algorithms, instead find correlations between words (or, more generally, short sequences of words called n-grams) in document text and subjects, based on a large amount of training data. These two approaches can be considered complementary, and often the best results are obtained by combining results from both kinds of algorithms using *ensembles* and/or *fusion architectures* (Toepfer & Seifert, 2018).

2.2.1. Lexical Approaches

Well-known lexical automated subject indexing systems include KEA and its successors KEA++ and Maui (Medelyan, 2009). They support multiple languages and they can be used with any indexing vocabulary. Another lexical automated subject indexing system is the Medical Text Indexer developed by the US National Library of Medicine for indexing medical documents with the Medical Subject Headings vocabulary (Mork, Jimeno-Yepes, & Aronson, 2013). Although all of these systems include some machine learning aspects, they are primarily based on lexical matching between vocabulary terms and document terms.

2.2.2. *Machine Learning Approaches*

Many machine learning based systems for automated subject indexing have been developed since the 1990s, when the approach became the dominant paradigm for automated subject indexing (Sebastiani, 2002). Some recent examples for which an implementation is available include Magpie (Berger, 2015; Kim, 2014), FastXML (Prabhu & Varma, 2014), PD-Sparse (Yen, Huang, Zhong, Ravikumar, & Dhillon, 2016), fastText (Joulin, Grave, Bojanowski, & Mikolov, 2017), Quadflor (Galke, Mai, Schelten, Brunsch, & Scherp, 2017), AnnexML (Tagami, 2017) and Parabel (Prabhu, Kag, Harsola, Agrawal, & Varma, 2018).

2.3. Web Services

Some automated subject indexing systems are also available as web services which can be integrated with existing document management systems. The BioPortal ontology repository[3] provides a service called Annotator, which is given text and matches terms in the text to concepts in biomedical vocabularies such as SNOMED CT. It can be used to find concepts in ontologies that correspond to variables or other entities in a description of a biomedical data set (Jonquet, Shah, & Musen, 2009). Another somewhat similar annotation and entity extraction service is DBpedia Spotlight, which finds occurrences of DBpedia entities within text (Daiber, Jakob, Hokamp, & Mendes, 2013). The focus of both of these tools is to provide a web API for retrieving all the matches within the given text. They do not try to determine the most relevant subjects of a document, so by themselves they only solve the first part of an automated subject indexing task, which is to determine possible candidate subjects for a given text.

2.4. Commercial Tools

Some commercial vocabulary management tools also include entity extraction and/or automated subject indexing functionalities. The PoolParty thesaurus management platform includes the PoolParty Extractor module,[4] which finds entities in text that correspond to concepts in the thesaurus and may also suggest new concepts for addition. This functionality was at least initially based on a modified version of KEA (Schandl & Blumauer, 2010). The TopBraid Enterprise Data Graph suite includes the AutoClassifier module[5] which uses Maui to perform a similar function, suggesting concepts that best represent the topic of an input document (Cyganiak, 2015). Both tools provide a REST-style web API that enables integration of the subject indexing functionality with other systems.

2.5. Limitations of Existing Systems

From the perspective of libraries, the systems mentioned above suffer from one or more drawbacks. First, many tools are limited to a single language (often English) and/or tied to a specific subject vocabulary. Yet subject indexing practices vary in different institutions and often there is a need to index materials in multiple languages and also use several different vocabularies. Second, the tools that are not language- or vocabulary-specific, including KEA, KEA++, Maui and the many machine learning algorithms, can be difficult to integrate with existing systems used for cataloguing and indexing, since they are either provided as command line tools or as software libraries in a particular implementation language. The commercial systems provide web services that are designed to be easily integrated with other systems, but their implementation is controlled by the respective vendors.

3. Architecture

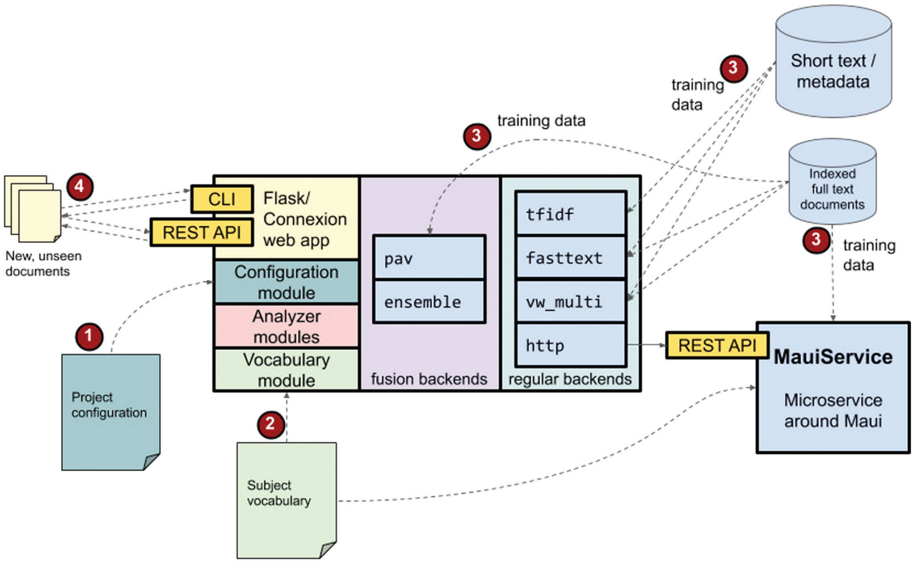
The first prototype of Annif was created in early 2017. It consisted of a loose collection of Python scripts that implemented a minimal REST API and a simple web user interface. An Elasticsearch index was used to find associations between subjects in a vocabulary and words in document titles that had been collected from the Finna API.[6] The idea was to turn a traditional text index on its head: instead of entering a topic and getting a list of documents about that topic in response, the input would be a single document and the output would be the most relevant topics for that document. The name given to the tool reflects this idea: Annif is Finna spelled backwards. The prototype worked well enough to demonstrate the utility of the approach, but the technical implementation would have been difficult to maintain, so development of a new version was started in early 2018.

The new Annif is implemented as a Python application using the Flask and Connexion frameworks for web server and REST API functionality. The subject indexing and classification is handled by different backends, i.e. different algorithms that can be used either alone or in combinations (so-called ensembles). Each backend is implemented as a separate module and new backends can be added in the future. A more detailed system architecture diagram is shown in Figure 1. This figure will be explained in more detail through sections 3.1 to 3.7.

3.1. Project Configuration

Annif is configured by defining *projects*, which are used to set up backends and configure them with a specific vocabulary and parameters, including its language and analyzer (see section 3.2 below). Each project is independent of

Fig. 1: Annif system architecture.



other projects but in some cases projects may be linked, for example by using the output of one project as the input of an ensemble project. The projects are defined in a configuration file (① in Figure 1).

3.2. Analyzers

Document text needs to be preprocessed before it can be analyzed with subject indexing algorithms. In Annif, text preprocessing is handled by Analyzer modules, that tokenize the text into sentences and individual words. Words may further be normalized using language-specific stemming or lemmatization algorithms. Tokenization and stemming are implemented using the NLTK library,[7] which provides a Snowball stemmer that supports 15 different languages. There is also a lemmatizing algorithm for the Finnish language based on the Voikko library.[8]

3.3. Vocabulary Support

Annif needs to be aware of the subject vocabulary that will be used for indexing. The vocabulary module handles loading and storing of vocabulary data. Vocabularies can be loaded either from simple TSV files or from SKOS/RDF files (② in Figure 1). The same vocabulary may be shared by multiple Annif projects and needs to be loaded only once.

3.4. Subject Indexing Algorithms

Currently, four subject indexing algorithms have been implemented as Annif backends. All implementations are based on existing open source libraries, which have been integrated into the Annif framework. *Maui* uses a lexical approach, while *TF-IDF*, *fastText* and *Vowpal Wabbit* represent different kinds of associative approaches. Most algorithms need to be trained using existing metadata and/or full text documents (③ in Figure 1).

3.4.1. Maui

Maui is a lexical automated subject indexing algorithm developed at the University of Waikato (Medelyan, 2009). It incorporates a large number of heuristics for determining which of the possible matches between terms in a vocabulary and words in a document best represent the topics of that document. The balance between the available heuristics is determined using machine learning, so a relatively small amount of manually indexed documents (up to a few hundred) is required for training a Maui model. In Annif, Maui is used via the *http* backend, which allows integrating subject indexing services which have a suitable REST API, including Annif itself and MauiService,[9] a REST microservice wrapper around Maui.

3.4.2. TF-IDF

The *tfidf* backend in Annif is a relatively simple statistical method used for finding correlations between subjects in the vocabulary and words in documents. A representative set of text is formed for each subject in the vocabulary by concatenating text (usually only titles) from documents that have been manually indexed with that subject. The *term frequencies* and *inverse document frequencies* are then calculated for all words appearing in those sets and these TF-IDF values are stored as vectors in an index. For new documents, TF-IDF vectors are similarly calculated and the most similar subjects are retrieved from the index. The calculations are performed using the Gensim library (Řehůřek & Sojka, 2010).

3.4.3. fastText

fastText (Joulin et al., 2017) is a machine learning algorithm for text classification created at Facebook Research. It claims to be roughly on par with deep learning approaches despite using a simpler architecture that resembles a shallow feed-forward neural network. The algorithm is relatively fast to train compared to other machine learning approaches, in part thanks to some tricks and shortcuts used in the implementation. The *fasttext* backend in Annif is a thin wrapper around the fastText Python bindings. There are quite a few hyperparameters to select and these may be tuned to attain good classification accuracy using a particular vocabulary and document corpus.

3.4.4. Vowpal Wabbit

Vowpal Wabbit (VW)[10] is a general purpose online machine learning framework. It was originally created by Yahoo! Research and current development continues at Microsoft Research. The *vw_multi* backend in Annif is a wrapper around several VW algorithms for multi-class and multi-label classification. Thanks to the online learning approach, the VW models can be further trained during use based on feedback from a user verifying the suggestions made by the algorithm. As the most recent addition to the Annif backends it has not yet been thoroughly evaluated, but it appears to be best suited for classification with relatively small (fewer than 1,000 classes/subjects) vocabularies.

3.5. Ensembles and Data Fusion

All automated subject indexing algorithms have their drawbacks. Incorrect subject assignments can be caused by many factors, including homonyms (e.g. 'rock' can mean stone or a kind of music), misinterpreted names (e.g. 'Smith' as a surname or a profession), correlations in data that do not imply causation, biased training data and random noise. Generally speaking, different kinds of algorithms tend to make different mistakes. A good strategy for improving quality is thus to combine different algorithms aiming to bring out the strengths of individual algorithms while diminishing their flaws.

Fusion methods for automated subject indexing (Toepfer & Seifert, 2018) are ways of combining results from multiple algorithms. The algorithms are combined into an ensemble and the final prediction of subjects is made by using a *decision function* applied on the predictions of individual algorithms. Fusion methods can be further divided into *descriptor-invariant* and *descriptor-specific* decision functions. In a descriptor-invariant function, every concept is handled in the same way, while descriptor-specific functions vary per individual concept. Annif supports two fusion backends, which combine results from configured source backends.

3.5.1. Simple Ensemble

The *ensemble* backend in Annif implements a simple, descriptor-invariant fusion method where the predictions from individual algorithms are merged by calculating the mean of score values for each predicted subject and using those as the final prediction. No learning is involved in this method.

3.5.2. PAV Ensemble

The *pav* backend in Annif implements a more advanced, descriptor-specific fusion method. It requires some more manually indexed full text documents for training in addition to those used to train the original backends. In experiments described in more detail in section 4 below, we have obtained good results using thousands of training documents to train PAV ensembles.

The training documents are first passed to the backend algorithms within the ensemble. Their prediction results are compared with the manually assigned subjects using *isotonic regression*, which is a statistical method that can be used for estimating the relationship between score values returned by the backends for particular subjects and the probability of the subject being relevant for the document (Wilbur & Kim, 2014). A separate regression model is created for each backend and each subject. New documents are first analyzed by the backends and the regression models are applied to the predicted scores, giving predicted probabilities. The final prediction is calculated using the mean value of the predicted probabilities.

3.6. Command Line Interface

Annif provides a command line interface which is mainly intended for initial setup, training, and evaluation of models. The training of models is done by providing Annif with training documents expressed in simple text file formats.[11] It can also be used to assign subjects to individual documents or document collections stored as text files. The command line interface can also be used to evaluate the algorithms by comparing their output to manually indexed document collections. Annif can be used to calculate many evaluation metrics, including precision, recall, F1 score and normalized discounted cumulative gain (NDCG).

When using the command line, the models need to be loaded from disk into memory separately for each invocation, so using large models is not very efficient. After initial setup and experimentation, setting up Annif as a persistent web service is recommended.

3.7. REST API

When Annif is run as a web service it provides a relatively simple REST API[12] which exposes the automated indexing functionality to other applications. The web server functionality of Annif is based on the Flask and Connexion toolkits and can be integrated with standard web server software such as Apache HTTPD using a WSGI gateway service (e.g. mod_wsgi). The core method of the API is *suggest*, which is given a text document and returns a JSON-encoded list of suggested subjects (concept URIs and labels) along with their estimated scores. Another important method is *learn*, which is given one or more text documents along with verified subjects for each document, and the corresponding models are updated based on this feedback. Currently only the Vowpal Wabbit based backend supports this kind of feedback-based online learning but learning support will be extended to other backends in the future.

4. Evaluation

Annif has been evaluated with several Finnish language corpora.

4.1. Vocabulary

All of the documents have been manually indexed using either the General Finnish Thesaurus YSA or its successor, the General Finnish Ontology YSO. For corpora indexed using YSA, the YSA subjects have been converted to their nearest YSO equivalents.

4.2. Training Data

The following algorithms were used:

- TF-IDF model trained using metadata from Finna.fi
- fastText model trained using metadata from Finna.fi
- Maui model trained using a combination of all the *maui-train* subsets
- PAV specific: PAV models trained on a *train* set specific to each corpus
- PAV generic: a single PAV model trained on a combination of all *train* sets

4.3. Document Corpora

The following corpora were used for evaluation:

1. **Arto**: Articles from the Arto[13] bibliographic database (n=6,287 articles). These include both academic articles as well as less formal publications from e.g. professional journals, and cover many different disciplines.
2. **JYU Theses**: Master's and doctoral theses from University of Jyväskylä (n=7,400) published in the years 2010 to 2017 (inclusive). These are long, in-depth academic documents that cover many disciplines.
3. **AskLib**: Question and answer pairs from the Ask a Librarian service run by public libraries in Finland. The original database consisted of over 25,000 documents but we extracted the subset with a minimum of 4 subjects per document (n=3,150). These are short, informal questions and answers about many different topics.
4. **Satakunnan Kansa**: Digital archives of Satakunnan Kansa regional newspaper. The archives consist of over 100,000 unindexed documents. Out of these, a random sample of 50 documents was manually indexed by four librarians working independently.

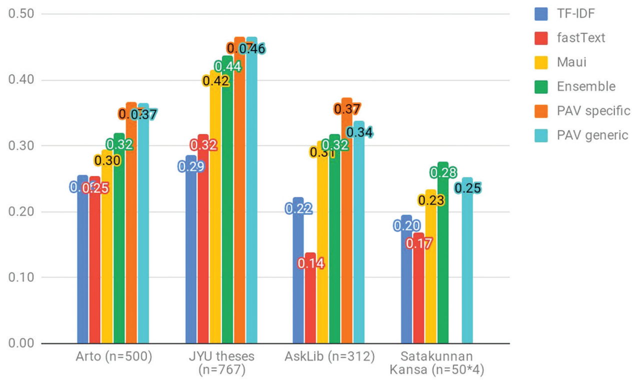
The corpora 1–3 are available on GitHub, in the Annif-corpora[14] public repository. In corpora 1 and 2, only links to PDF files are provided due to copyright reasons, but the full text is available elsewhere on the web.

Each corpus was split into *train*, *validate* and *test* subsets, where the *train* set was to be used for training Annif algorithm, the *validate* set for choosing optimal hyperparameters and limit/threshold settings, and the *test* set for final evaluation. For the Arto corpus, a random split was used. For JYU Theses and AskLib, the corpus was split by publication time: documents published before 2016 were assigned to the *train* set, documents published in 2016 to the *validate* set and documents published in 2017 to the *test* set. For the first three corpora, an additional *maui-train* subset, intended for training Maui models, was created by taking a random sample of 200 documents from the *train* set. For Satakunnan Kansa, all the manually indexed documents were used only as a *test* set. Each document was indexed independently by four librarians. We considered each set of subjects independently, so the evaluation was performed on 200 different document/subject combinations. The number of documents in each subset are summarized in Table 1.

Table 1: Subsets of the document corpora used for evaluation.

Corpus	**# train**	**# maui-train**	**# validate**	**# test**
Arto	5,287 (84%)	200 (3%)	500 (8%)	500 (8%)
JYU Theses	3,635 (70%)	200 (4%)	786 (15%)	766 (15%)
AskLib	2,625 (83%)	200 (6%)	213 (7%)	312 (10%)
Satakunnan Kansa	–	–	–	50*4

Fig. 2: Evaluation results.



4.4. Evaluation Results

The main evaluation measure was F1 score. However, since F1 score is sensitive to the number of subjects assigned to each document, which is affected by the limit and threshold parameters, we applied a limit of at most suggested 5 subjects per document, which appeared to be near-optimal based on evaluation on the *validate* sets. The results of evaluations on the final *test* sets are shown in Figure 2.

Based on the evaluation results, we can conclude the following:

1. Of the individual algorithms, Maui performed best on all corpora. The relative performance of TF-IDF and fastText varied by corpus, with TF-IDF being somewhat better on average.
2. The ensemble models were always superior to individual algorithms.

3. The PAV ensembles were generally superior to plain ensembles, with the exception of the Satakunnan Kansa corpus.
4. The generic and specific PAV ensembles were roughly on par, but for AskLib, the specific PAV ensemble performed slightly better.

5. Usage Scenarios

Automated subject indexing can be used to assist manual indexing (*semi-automated indexing)*, so that an algorithm is used to suggest subjects for a new document which are then verified manually, or independently (*fully automated indexing)*, so that the suggestions of the algorithm are accepted without manual verification. Annif may be used in both kinds of scenarios as well as some less conventional settings.

5.1. Semi-Automated Indexing

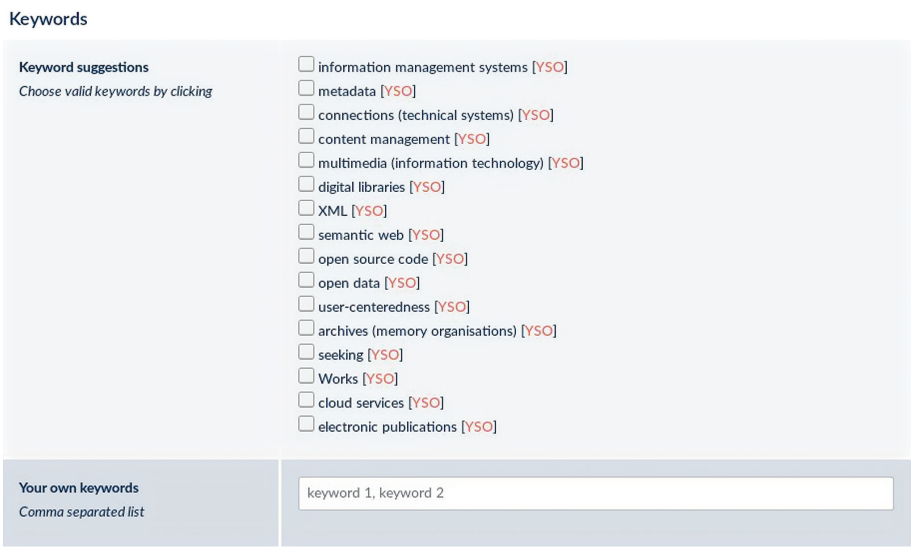
In semi-automated subject indexing, the quality of results is not as critical as in the fully automated case, but the suggestions of the algorithm must still provide value to the indexer instead of being a distraction. Automated suggestions can be incorporated into existing manual indexing workflows.

5.1.1. JYX Institutional Repository

The University of Jyväskylä has integrated Annif into its institutional repository JYX,[15] which is used, among other purposes, for archiving Master's and doctoral theses. Students upload their thesis to the repository as a PDF file and are then requested to enter metadata about the thesis, including subjects. The text is extracted from the PDF document and sent to the Annif REST API for analysis. The predicted subjects are shown to the student, who can then select the most appropriate subjects and also enter additional subjects that the algorithm has missed. A screenshot of the suggestions is shown in Figure 3.

The university was an early adopter of Annif and started using the REST API of the Annif prototype in May 2018, when a new version of the JYX repository was launched. In the beginning of November 2018, JYX switched to the

Fig. 3: Subjects suggested by Annif after uploading a document to the JYX repository.



Keywords

Keyword suggestions *Choose valid keywords by clicking*	☐ information management systems [YSO] ☐ metadata [YSO] ☐ connections (technical systems) [YSO] ☐ content management [YSO] ☐ multimedia (information technology) [YSO] ☐ digital libraries [YSO] ☐ XML [YSO] ☐ semantic web [YSO] ☐ open source code [YSO] ☐ open data [YSO] ☐ user-centeredness [YSO] ☐ archives (memory organisations) [YSO] ☐ seeking [YSO] ☐ Works [YSO] ☐ cloud services [YSO] ☐ electronic publications [YSO]
Your own keywords *Comma separated list*	keyword 1, keyword 2

REST API of the new Annif implementation. The university has collected data about the subjects suggested by Annif for Master's theses, the choices made by students and the final subjects assigned by librarians, who perform the final validation of metadata. This data makes it possible to evaluate the quality of subjects suggested by Annif and to compare the quality of the Annif prototype against the new version.

From May to October 2018, 890 Master's theses were uploaded to JYX and analyzed by the Annif prototype. From November 2018 to January 2019, a further 385 Master's theses were uploaded and analyzed by the new version of Annif, which used a simple ensemble model combining TF-IDF, fastText and Maui algorithms.

Similarity between the subjects suggested by Annif (either the prototype or new version), the subjects selected by students and the final subjects assigned by librarians is shown in Figure 4. We can see that approximately one third of the subjects suggested by the Annif prototype were selected both by students and the librarians making the final choices, which already shows that the system provided value to the users of JYX. However, the results for the new version were much better: students selected approximately one half of the suggestions by Annif, and the librarians slightly more (53%). The variation in F1 scores between documents is quite high, as shown by the error bars, indicating that the results were much better for some theses than for other. In the case of students, some of this variation can be explained by students who did not select any subject from the suggestions (30% for the prototype, 15% for the new version). We cannot tell whether this happened because the suggestions were very bad or because of some other, unrelated reason.

The similarity scores for the new version are analyzed broken down by university department in Figure 5. Due to the relatively small number of documents and the high variation in F1 scores, we cannot draw any firm conclusions, but it appears that the best results are obtained in the humanities, while results are not as good in mathematics, science and technology. This pattern may be due to differences in granularity of the subject vocabulary in

Fig. 4: F1 similarity between Annif suggestions, student-selected subjects and final subjects in JYX, for the Annif prototype and new version.

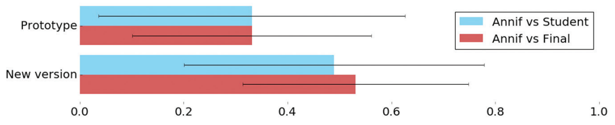
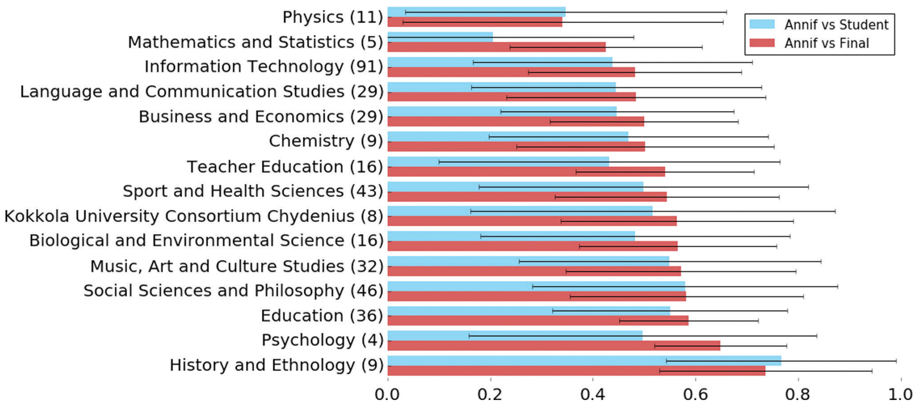


Fig. 5: F1 similarity between Annif (new version) suggestions, student-selected subjects and final subjects in JYX, by department.



different topical areas, as well as the different nature of concepts in different fields: in the humanities, concepts may often be broader and fuzzier, whereas in more technical fields they can be more specific and strictly bounded.

5.2. Fully Automated Indexing

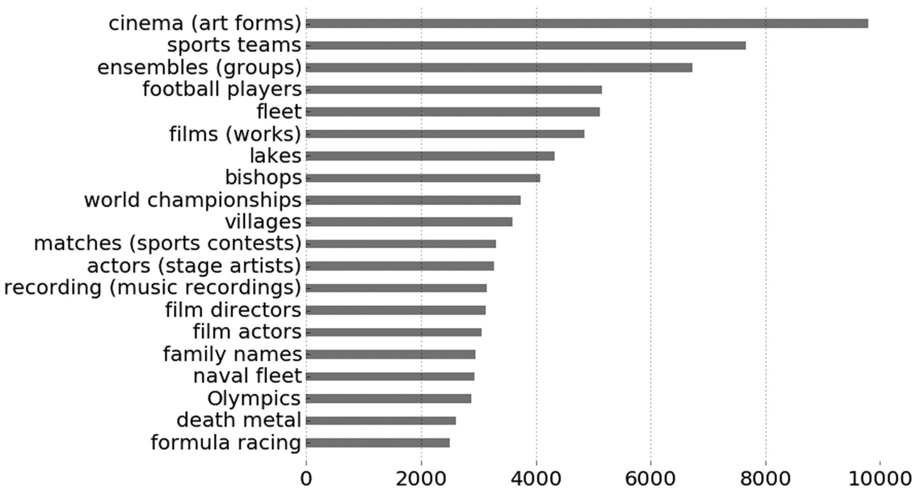
Fully automated indexing is suitable for large document collections, where manual verification of suggested subjects is not feasible. Typically, stricter criteria are applied on the suggested concepts: the number of subjects per document is limited to a small number and a high score or probability threshold is used to restrict the assigned subjects to only the most certain ones. To demonstrate how Annif can be applied for automatically indexing large document collections, we have tested it on two large document corpora: Finnish Wikipedia and the digital archives of Satakunnan Kansa regional newspaper.

5.2.1. Finnish Wikipedia

We downloaded the full database dump of Finnish Wikipedia articles dated 2019-03-01 and converted it to plain text using the WikiExtractor tool.[16] The dump included 452,857 articles. Each article was analyzed with Annif using a simple ensemble consisting of TF-IDF, fastText and Maui backends. Relatively strict criteria were used for selecting subjects, both because Wikipedia articles are focused on a single topic and to avoid false positives that could skew the analysis. A maximum of 3 subjects per article were chosen, and a score threshold of 0.85 relative to the best score was used (i.e., if the best subject got a score of 0.5, then up to two other subjects with a score of at least 0.425 were included as well). This resulted in 1.56 subjects per article on average. The processing was performed on a standard virtual server using four CPU cores in parallel and took about 16 hours, at a rate of 8.0 articles per second. The most frequently occurring subjects according to this analysis are shown in Figure 6.

If we group the top 20 subjects by themes, we can see that the most common themes include cinema (films, actors, directors), music (musical groups, music recordings, death metal), sports (football, world championships, sports matches, Olympics, formula racing), and geography (lakes, villages). Some

Fig. 6: Most frequently occurring subjects in Finnish Wikipedia articles.



surprisingly common themes are subjects related to the navy (e.g. fleet and naval fleet) and bishops. A spot check of articles indexed with these subjects reveals that there really are quite many pages about individual warships on Finnish Wikipedia, as well as biographical pages for bishops, most of them apparently imported from a database of Catholic priests. The analysis gives a thematic overview of Finnish Wikipedia that would be difficult to obtain using text processing (e.g. calculating word frequencies) alone. Since the vocabulary YSO is trilingual, the same analysis could also potentially be performed on Swedish and English Wikipedia and the results compared on a conceptual level.

5.2.2. Satakunnan Kansa

We performed a similar analysis using the same methods as with the Finnish Wikipedia articles on the digital archives of the Satakunnan Kansa regional newspaper, which contains 111,850 articles published between 1987 and 2004. The analysis took about 4.5 hours, or 7.1 articles per second. In this case, the main themes were related to municipal decision-making and development (e.g. municipal councils, local executives, chairpersons, schools, plots of land, municipal managers), use of money and currencies (Finnish markka, euros, budgets) and the local jazz music festival Pori Jazz. However, many articles were incorrectly assigned subjects related to specific buildings such as the Pori Orthodox Church, the Church of Holy Trinity in Rauma, and the Vanhakartano Manor in Köyliö. The articles indexed with those subjects were mostly not concerned with those buildings but were more generally about the cities of Pori and Rauma and the former municipality of Köyliö. However, since the YSO subject vocabulary does not include places—they are in a separate vocabulary called YSO Places—the algorithms ended up suggesting buildings located in those places instead. Even in this case the analysis gives a thematic overview of the newspaper archives, but the results need to be interpreted carefully as some of the assigned subjects can be misleading.

5.3. Unconventional Uses

While semi-automated and fully automated indexing are the main usage scenarios of Annif, it can also be used for novel purposes. Since Annif provides a simple REST API, it can be easily integrated into various tools that go beyond the scope of traditional automated subject indexing.

5.3.1. Supporting Indexing of Printed Materials

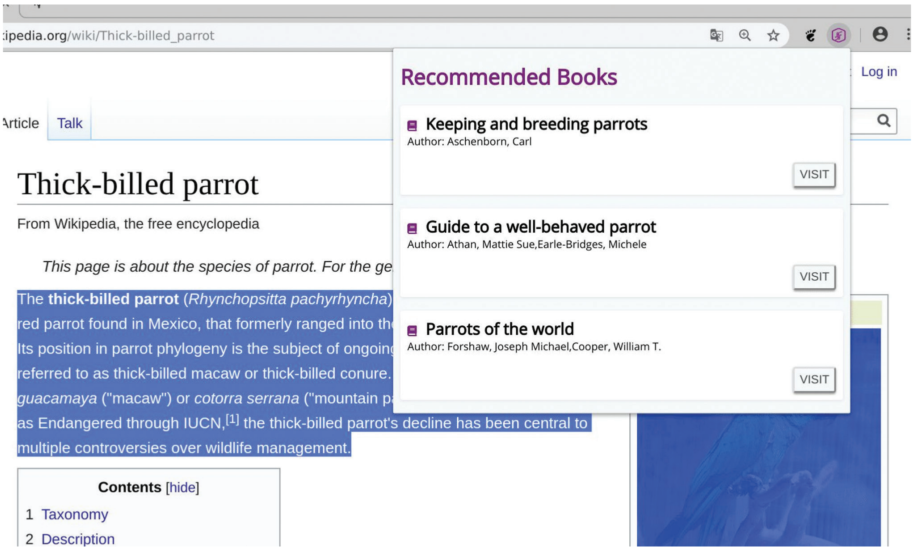
Although automated indexing is mostly applied to digital materials, we have explored possibilities to use Annif for assisting in the indexing of traditional printed materials such as books and articles. We have built two prototype mobile apps that use the camera in a tablet or smartphone to take a picture of a document (or a part of it such as the introduction section), convert it to text using optical character recognition (OCR) technology, and analyze it using the Annif REST API.

The first prototype[17] is a mobile web application that runs within the browser of a mobile device. It uses a cloud OCR service to convert the picture into text, which is relatively slow because the picture needs to be uploaded to the web, but the app works on any mobile device with a modern browser. The second prototype is a native Android app which uses the Google ML Kit[18] library to perform real time OCR on the mobile device. In both apps, the user will then be presented with a list of suggested subjects, usually in a much shorter time than it would take to read the document. However, these prototypes are currently just demonstrations of the idea and we have not yet performed any formal testing of these apps as part of an actual subject indexing workflow.

5.3.2. Recommending Documents Based on Web Page Text

One of the applications developed at a hackathon organized by the National Library of Finland, together with other partners, was a Chrome browser extension called Finna Recommends.[19] The extension adds a small button with the Finna icon to the browser toolbar. The user can select any text from a web page and then click the button to get recommendations of related books (see Figure 7). Behind the scenes, the selected text is given to the Annif API, then the top three subjects suggested by Annif are used to query for books in the Finna API. This extension makes the collections of libraries available to any web user using just a single click, without the user having to think about suitable keywords.

Fig. 7: The Finna Recommends browser extension suggests of books based on selected web page text. The user has selected some text on a Wikipedia page for a parrot species and is shown recommendations for books about parrots.



Recommended Books

Keeping and breeding parrots
Author: Aschenborn, Carl

Guide to a well-behaved parrot
Author: Athan, Mattie Sue,Earle-Bridges, Michele

Parrots of the world
Author: Forshaw, Joseph Michael,Cooper, William T.

Thick-billed parrot

From Wikipedia, the free encyclopedia

This page is about the species of parrot. For the ge

The **thick-billed parrot** (*Rhynchopsitta pachyrhyncha*) red parrot found in Mexico, that formerly ranged into th Its position in parrot phylogeny is the subject of ongoin referred to as thick-billed macaw or thick-billed conure. *guacamaya* ("macaw") or *cotorra serrana* ("mountain p as Endangered through IUCN,[1] the thick-billed parrot's decline has been central to multiple controversies over wildlife management.

5.3.3. Powering a Chatbot

We have created a prototype chatbot user interface, called AnnifBot,[20] which asks questions about the user's interests, turns the responses into YSO subjects using the Annif REST API, and then looks up books and images indexed with those subjects from the Finna API. The functionality is similar to a more traditional search engine such as the main discovery user interface of Finna, but providing a conversational user interface instead of a search form. In the future, such a chatbot could be integrated into Finna or other similar systems to make them more engaging and interactive. A similar chatbot could also use a custom vocabulary and model which identifies frequently occurring user interests and provides appropriate answers.

6. Discussion and Conclusion

Libraries and related institutions have a clear need for automating some of their indexing workflows. For this they need practical tools that provide sufficient indexing quality and that can be integrated into existing systems. Some commercial tools are available, but they may not always be attractive due to their cost, limited vocabulary and/or language support, or the vendor lock-in aspect. While many open source automated subject indexing projects are available, they are generally implementations of individual algorithms which may not be easy to integrate with other systems. Annif provides a new alternative in this space and is designed to be extensible by adding new analyzers and subject indexing algorithms.

Annif is based on a combination of natural language processing and machine learning tools. Annif can be adapted to different settings, including both subject indexing and classification, and it can make the best use of the results from different analysers. In our initial evaluations, we have found that combinations of existing algorithms generally perform better than individual algorithms. Using an ensemble of several algorithms, we could beat the F1 score of Maui, which itself is advertised as achieving human-competitive indexing quality (Medelyan, 2019), by several percentage points on multiple very different document corpora.

Providing the Annif functionality as a REST API microservice makes it relatively easy to integrate automated subject indexing functionality into existing systems, as exemplified by the JYX institutional repository. We are planning to integrate more systems with Annif, including those used for receiving electronic deposits and for processing digitized materials. The API service also enables novel applications, including mobile apps, browser extensions and chatbots.

We are planning to further develop Annif by adding new backend algorithms and incorporating online learning support for more backends. We also aim to evaluate it with new corpora and different kinds of vocabularies, including place names and library classifications such as UDC and DDC. We expect to use Annif to help improve subject indexing and classification processes especially for electronic documents as well as collections that otherwise would not be indexed at all.

Acknowledgements

We thank Martin Toepfer for stimulating discussions around automated indexing methods, Markus Koskela for the suggestion to use ranking-based evaluation measures available in the scikit-learn toolkit, Sampo Savolainen for improving the Maui codebase and creating MauiService, Ari Häyrinen for being an early adopter of Annif and for collecting observations of its use in the JYX document repository, Okko Vainonen for developing the Annif Android app, and Yazan Alhalabi, Samuel Akangbe and Steven Nebo for creating the Finna Recommends browser extension. We thank Thomas Baker, Juho Inkinen, Anna Kasprzik, Bruno P. Kinoshita, Riitta Koikkalainen, Mona Lehtinen, Tuula Pääkkönen, and Hugo de Vos, who provided insightful and constructive comments on the draft manuscript.

References

Berger, M.J. (2015). *Large scale multi-label text classification with semantic word vectors. Technical Report.* Stanford University. Retrieved July 8, 2019, from https://cs224d.stanford.edu/reports/BergerMark.pdf.

Cyganiak, R. (2015, September 22). Deep dives into TopBraid EVN — Part 1: Automated tagging with the New AutoClassifier. *The Semantic Ecosystems Journal*. Retrieved March 29, 2019, from https://www.topquadrant.com/2015/09/22/automated-tagging-evn-autoclassifier/.

Daiber, J., Jakob, M., Hokamp, C., & Mendes, P.N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In M. Sabou, E. Blomqvist, T. Di Noia, H. Sack & T. Pellegrini (Eds.), *Proceedings of the 9th International Conference on Semantic Systems* (pp. 121–124). New York: ACM. https://doi.org/10.1145/2506182.2506198 Open access copy available at http://informatica.uniroma2.it/upload/2018/IA2/Improving%20efficiency%20and%20accuracy%20in%20multilingual%20entity%20extraction.pdf.

Galke, L., Mai, F., Schelten, A., Brunsch, D., & Scherp, A. (2017). Using titles vs. full-text as source for automated semantic document annotation. In *Proceedings of the Knowledge Capture Conference (K-CAP 2017)* (pp. 20:1–20:4). New York: ACM. https://doi.org/10.1145/3148011.3148039 Open access copy available at https://arxiv.org/pdf/1705.05311.

Jonquet, C., Shah, N.H., & Musen, M.A. (2009). The open biomedical annotator. *Summit on Translational Bioinformatics*, *2009*, 56–60. Retrieved July 8, 2019, from https://www.ncbi.nlm.nih.gov/pubmed/21347171.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In M. Lapata, P. Blunsom, & A. Koller (Eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Volume 2, short papers* (pp. 427–431). Stroudsburg, PA: ACL. Retrieved July 8, 2019, from http://aclweb.org/anthology/E17-2068.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181.

Medelyan, O. (2009). *Human-competitive automatic topic indexing*. Doctoral thesis, University of Waikato, Hamilton, New Zealand. Retrieved July 8, 2019, from https://hdl.handle.net/10289/3513.

Mork, J.G., Jimeno-Yepes, A., & Aronson, A.R. (2013). The NLM medical text indexer system for indexing biomedical literature. In *BioASQ@ CLEF*, *Proceedings of the first workshop on Bio-Medical Semantic Indexing and Question Answering, a post-conference workshop of Conference and Labs of the Evaluation Forum 2013 (CLEF 2013)(n.p.).* Retrieved July 8, 2019, from https://ii.nlm.nih.gov/Publications/Papers/MTI_System_Description_Expanded_2013_Accessible.pdf.

Pouliquen, B., Steinberger, R., & Ignat, C. (2003). Automatic annotation of multilingual text collections with a conceptual thesaurus. In *Proceedings of the Workshop 'Ontologies and Information Extraction' at the EUROLAN Conference, Cluj-Napoca, Romania* (pp. 19–28). Retrieved from https://arxiv.org/abs/cs/0609059.

Prabhu, Y., & Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 263–272). New York: ACM.

https://doi.org/10.1145/2623330.2623651. Open access copy available from http://manikvarma.org/pubs/prabhu14.pdf.

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., & Varma, M. (2018). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web* (pp. 993–1002). International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/3178876.3185998.

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 46–50). University of Malta. Retrieved July 8, 2019, from https://is.muni.cz/publication/884893/en.

Schandl, T., & Blumauer A. (2010) PoolParty: SKOS thesaurus management utilizing linked data. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, & T. Tudorache (Eds.), *The Semantic Web: Research and Applications*. ESWC 2010. Lecture Notes in Computer Science, vol 6089 (pp. 421–425). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-13489-0_36.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283. Retrieved from https://arxiv.org/abs/cs/0110053v1.

Tagami, Y. (2017). AnnexML: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 455–464). New York: ACM. https://doi.org/10.1145/3097983.3097987.

Toepfer, M., & Seifert, C. (2018). Fusion architectures for automatic subject indexing under concept drift. *International Journal on Digital Libraries*, 1–21 (E-pub ahead of print). https://doi.org/10.1007/s00799-018-0240-3 Open access copy available from https://research.utwente.nl/files/80439235/Toepfer2018_ijdl_subject_indexing_under_concept_drift_preprint.pdf.

Wilbur, W.J., & Kim, W. (2014). Stochastic gradient descent and the prediction of MeSH for PubMed records. *AMIA Annual Symposium Proceedings*, *2014*, 1198–207. Retrieved July 8, 2019, from https://www.ncbi.nlm.nih.gov/pubmed/25954431.

Yen, I.E.H., Huang, X., Zhong, K., Ravikumar, P., & Dhillon, I.S. (2016). PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In M.F. Balcan & K.Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on Machine Learning* (pp. 3069–3077). New York: ACM. Retrieved July 8, 2019, from http://proceedings.mlr.press/v48/yenb16.pdf.

Notes

[1] See https://github.com/NatLibFi/Annif.

[2] https://doi.org/10.5281/zenodo.2578948.

[3] See https://bioportal.bioontology.org/.

[4] See https://www.poolparty.biz/poolparty-extractor/.

[5] See https://www.topquadrant.com/products/topbraid-tagger-autoclassifier/.

[6] See https://api.finna.fi.

[7] See http://www.nltk.org/.

[8] See https://voikko.puimula.org/.

[9] See https://github.com/NatLibFi/mauiservice.

[10] See http://hunch.net/~vw/ and https://github.com/VowpalWabbit/vowpal_wabbit.

[11] See https://github.com/NatLibFi/Annif/wiki/Document-corpus-formats for documentation about formats.

[12] See http://api.annif.org for API documentation.

[13] https://www.kansalliskirjasto.fi/en/services/metadata-reserve-services/arto.

[14] https://github.com/NatLibFi/Annif-corpora.

[15] See https://jyx.jyu.fi/.

[16] See https://github.com/attardi/wikiextractor.

[17] See http://m.annif.org.

[18] See https://developers.google.com/ml-kit/.

[19] See https://github.com/YazanAlhalabi/Finna-recommends.

[20] See http://bot.annif.org.