Modular Development in Patent AI Space: A Case Study

Mahesh Maan
Data Science Team, GreyB
Mohali, India
mahesh@projectpq.ai

Sam Zellner
Project Lead, PQAI
Atlanta, USA
sam@projectpq.ai

Anirudh Sanutra
IP Solutions, GreyB
Mohali, India
anirudh@projectpq.ai

ABSTRACT

Numerous development efforts are underway that aim to apply the recent advancements in the field of artificial intelligence to a variety of patent-related tasks such as prior-art searching, technology landscaping, patent classification, etc. In this paper, we advocate for researchers to align their work with a modular system of software components. We show how such alignment will make it easier for researchers to prototype new systems, upgrade existing systems, collaborate, and build upon each other's work. We also present such a system of modular components that we created while developing PQAI, an AI-based prior-art search platform.

KEYWORDS

Patent AI, Patent Retrieval, Prior Art Search, Open Source, Modular Software Architecture

1 Introduction

As a result of the rapid advancements in the field of machine learning, the prevalent optimism surrounding AI in general, and the ongoing efforts in IP industry [1, 2, 3, 4, 5, 6] and academia [7, 8, 9, 10], we foresee the development of many AI-centered patent data mining software platforms in the coming years. If the current trend persists, the field of AI will continue to evolve rapidly during this time. As a result, to stay relevant, these platforms will need to continuously experiment with and adopt the latest and best AI tools as they become available.

In this paper, we show that by designing modular components, researchers involved in the development and refinement of such platforms can ensure that their systems are flexible enough to evolve quickly and versatile enough to leverage a spectrum of AI technique. We advocate for more researchers to consider fitting their work within a modular system. This would enable the community of patent data mining researchers to more easily collaborate and build upon each other's work.

We also present an example of such a modular system, designed during the development of PQAI [11], an AI-based prior-art search platform.

The structure of this paper is as follows: Section1 sets the context of the paper. Section 2 highlights a number of existing problems associated with the development of AI-centric patent-data mining platforms (and which might become more critical in future as these platforms continue to develop in isolation). Section 3 describes how a common, open-source schema of modular components could mitigate these problems. Section 4 delves into the details of the modular system used in PQAI and describes some of its components. Section 5 presents few examples that show such components could be interlinked to build modular systems to target specific patent data mining problems.

2 Problems

Currently, most research groups developing patent-AI platforms work in isolation. This is imparting to a number of inefficiencies in the development efforts and creating limitations for the users of such tools too. We are highlighting them below:

1. Although each platforms has a unique proposition that focuses on a specific AI-capability, to render it usable in the form of a software tool, a number of 'commodity' software components for handling patent data also need to be developed. Such components include patent number parses, patent data repository wrappers, patent reading interfaces, exporting and reporting functions, etc. Each development group has to spend considerable resources building commodity components that do not add to their core value proposition.
2. The fact that there is no standardization of patent-software components makes it difficult for various groups to collaborate, upgrade, license, or sell their components for reuse by others. Even though some of the platforms make few API services available, they are also mostly incompatible with each other, making it difficult to merge the data/functionalities of two systems.
3. The closed source implementation of most platforms means that it is not feasible for users such as corporate IP teams and law-firms, who would like to have a custom UI or some custom functionality on top of what is available on the platform by default.
4. Since most platforms do not provide access to their inner components or APIs, the entry-effort for AI researchers who want to explore and experiment with patent data (e.g.,

in universities) is high. They have to arrange their own databases and create a base functionality layer as a pre-requisite. This takes considerable time and effort.

3 Solution

To mitigate the problems described in the preceding section, we propose a solution centered on development, dissemination, and adoption of an open-source schema of components with standardized interfaces and their reference implementations by the research community working in the patent-AI space.

Our solution is inspired partly by few initiatives [12, 13, 14, 15] that facilitate modular and efficient development in neighboring fields and partly by our own exploration of the design space of a patent-AI system while developing a prior-art search engine. Although modular development in patent information retrieval has been attempted before [16, 17], in practice, there is a general lack of open-source frameworks and resources critical for driving industry-wide adoption and efficiency.

We believe that the availability of such resources would help researchers to develop experimental prototypes faster, upgrade existing software platforms with ease, and smoothly build upon each other's work. These resources include:

1. *An open-source schema of standardized software components*: This schema acts as a blueprint of a family of highly customizable and frequently used software components that can be inter-connected together like Lego pieces. These components should be standard in the sense that they have well-defined input-output characteristics, although the schema imposes no restriction on their implementation. Following such a schema can help in ensuring that components created by groups completely isolated from each other are still inter-compatible. This schema should be defined at a high level to be able to accommodate a variety of AI techniques (few examples are presented in next section).
2. *Open-source implementations of software components:* Access to a library of such components will enable researchers and developers to avoid spending time creating their own 'base layer' functionalities (such as standard searching and filtering operations, patent number parsing, patent rendering, data management, etc.) and reference implementations for comparison. Instead, maximum effort could then be invested in creating new and improved components or building new capabilities by combining existing components in new ways.
3. *Remote API access to software components and datasets:* API access will facilitate and encourage small-scale development and experimentation with patent data. This would be useful for resource constrained efforts, such as where a small in-house IP team can hire a freelancer developer to create a lightweight dashboard for accessing and exploring patent data of their own specific field, or where a small team of students is carrying out a university project involving patent data analysis. The API access would obviate the need for such groups to set up a heavy system to get a little done.

4 Components

In this section, we present a non-exhaustive list of versatile components from the PQAI library [11], which offer functionalities frequently required in a range of patent data mining operations. Under the PQAI initiative, the authors aim to define a standardized schema of these components and release their concrete implementations as open-source code.

For some of the components, we first describe their abstract versions and then the patent-specific version. The abstract version is unlikely to be used in real-world development, but knowing its behavior helps in understanding the behavior of the family of components that can be derived from it.

Patent Database: This is an instance of abstract component Storage (described at the end of this list). Patent Database is a special component in the sense that all components can be configured to access it. A major benefit of this approach is that components can pass around references within the Patent Database (e.g., patent numbers) instead of patent data itself. This keeps the component interfaces clean and lightweight.

Encoder: An encoder takes in an *entity* and returns its *representation*. The input entities and output representations can both take many forms, making this component very versatile. For instance, one instantiation of an encoder can be in the form of a *Patent Vectorizer* – which accepts a patent number (as described earlier, all components can retrieve patent data given the patent number) and returns a vector embedding in a high dimensional space that corresponds to the given patent. A bag-of-words encoder can be another example of this component.

Index: An index is a data structure optimized for searching among entity representations. It differs from a Store in that it may not necessarily be able to return the original representation. It accepts a compatible query and returns a set of entity pointers. A *Patent Vector Index*, for example, might accept a query vector and return a set of patent number as top matches for the query. Note that the Index accepts query representations and not raw queries, therefore, it has to plugged into a suitable Encoder to turn the raw query into a compatible representation.

Ranker: It accepts a set of entities and returns a list of the same entities, the order of which is determined by a ranking criterion. A *Patent Ranker* for instance, would accept a set of patents and a user query as input and orders those patents in decreasing order of relevancy to the given query.

Classifier: A classifier associates one of a finite set of predefined labels to a patent, where the labels have unique meanings associated with them. A *Patent Classifier* for instance, could take as input a set of patent numbers and associate, with each patent number, a label, which may mean for example whether this patent is related to solar cell technology or not. Internally, classifiers can make use of configurable classifier models, which can be initialized with inputs such as (patent-number, label) pairs or a textual description.

Consolidator: A consolidator accepts a set of patents and associates one of a finite set of arbitrary labels to each patent. Essentially, it creates clusters of patents where each cluster's patents have some common characteristics. A *Technology Consolidator*, for instance, can accept a set of patent numbers and then group them into, say, 3 groups, depending on the technologies they relate to.

Filter: A filter accepts a set of entities and depending on a filter criterion returns a subset of them. A *Patent Filter*, for instance, would filter out patents satisfying a condition such as a publication date criterion. Filters can be cascaded to create a *Filter Sequence*. For instance, a date *period* filter can be created by cascading a before-date filter and an after-date filter.

Sorter: The input-output characteristics of a sorter are similar to a ranker but in its output, only the relative positions of the entities matter. A *Patent Sorter* can, for instance, accept a list of patent numbers and arrange them such that any patent in the list is succeeded by the most similar patent to it in the rest of the list. Such a sorter can be useful during a manual review of patents (all related patents come in sequence and the reviewer can make use of insights still available in their short term memory).

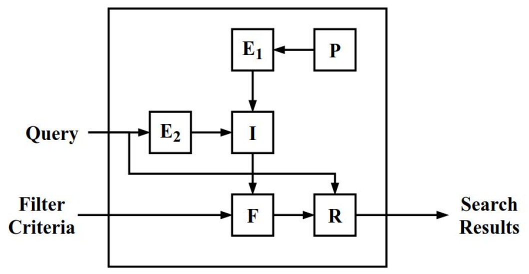
Patent Number Parser: It accepts plain text as input, then detects and extracts any patent numbers in it, translates them into a standard format (e.g., by truncating or adding zeros) and then outputs a list of patent numbers that can be directly inputted to the *Patent Database* component. This component, when used at the boundary of a patent data mining system, can eliminate all issues that arise due to patent number format mismatching.

Storage is an abstract wrapper around as a data source. It stores entities that are all of the same type but other than this, it makes no assumption about how the data is stored (e.g. whether it is stored in a local database, in the primary memory, or on a remote server). A Storage component performs two operations: it saves and retrieves entities. In the saving operation, it accepts an entity and returns its entity identifier. Retrieval operation is the opposite of saving operation - an entity is returned in response to a supplied identifier. Storages can be configured to be read-only too.

5 Exemplary Systems

In this section we show few examples to demonstrate how the components described in the last section can be interconnected to form systems that carry out specific patent-related tasks.

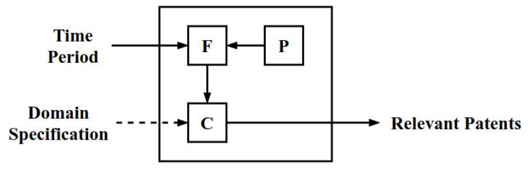
Patent search engine: The input here is a user query and possibly one or more filters (such as date restrictions) and the output is a ranked list of patents. The following diagram shows a possible implementation using standard components:



P: Patent Database, **E$_1$**: Patent Encoder, **E$_2$**: Query Encoder, **I**: Index (precomputed), **F**: Filter Sequence, **R**: Patent Ranker

Figure 1: A patent search engine designed with standard components

Technology monitoring system: The input here is a specification of a technology area and a time period of interest and the output is a set of recent patents and published applications. A generic, re-configurable classifier is used here, which uses a textual description of the technology or a white-list of patents to adjust its operation (e.g., by training an internal ML model). The diagram below shows a possible implementation using standard components:



P: Patent Database, **F**: Date Filter, **C**: Classifier

Figure 2: A technology monitoring system designed with standard components

Technology landscaping: The output of technology landscaping studies, unlike the preceding two examples, is much more complex than a list of patents. A typical output, however, can be broken down in the form of a number of insights that are reached by manually analyzing the patent data, typically through charts on a dashboard. Some of these charts are quite basic, such as patent filing trend over the years. Others are more sophisticated, such as a heat-map of patent portfolio sizes held by the major players in a number of technology sub-domains. Most of these charts, irrespective of the complexity of the underlying data, plot two categorical variables. The data for these charts can therefore, be arrived at by making use of two consolidators, each of which cluster the data points into discrete clusters which correspond to the plotted categories:

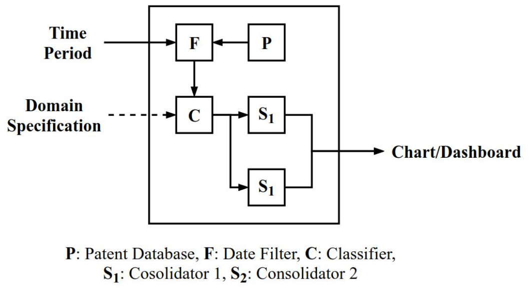

Time Period
Domain Specification
F
P
C
S1
S1
Chart/Dashboard
P: Patent Database, F: Date Filter, C: Classifier,
S1: Cosolidator 1, S2: Consolidator 2

Figure 3: Part of a technology landscaping system designed with standard components

6 Conclusions and Future Work

We showed how the ongoing and future software development efforts in the patent-AI space can be greatly facilitated by aligning the development with a schema of modular and highly-customizable software components. Through examples we showed how the components in such a schema can be interconnected like Lego bricks to create varied patent data mining systems optimized for different tasks. Following such a schema will enable faster prototyping, smoother upgrades, and easier collaboration among research groups. We therefore encourage researchers to consider fitting their work in a common modular system to speed up research and development in the patent-AI space.

In future work, we aim to define and release such a modular system and open-source implementations of its constituent components under the PQAI initiative. We invite industry experts and researchers for collaboration in defining and standardizing this system.

REFERENCES

[1] IPRALLY TECHNOLOGIES LTD, 2021. Patent search done right. https://www.iprally.com/
[2] AMPLIFIED, 2020. Better prior art. Faster. https://www.amplified.ai/
[3] SIMILARI, 2020. Supercharge your IP. https://similari.com/
[4] INQUARTIK CORPORATION, 2021. Find patents with ease. https://www.inquartik.com/patentcloud/patent-search/
[5] THREE10 SOLUTIONS, INC. 2020. The End of Keyword Searching. https://www.dorothyai.com/platform
[6] CINTIAN, 2020. Welcome to the Patent Data Revolution. https://www.cintian.ai/
[7] Helmers, L., Horn, F., Biegler, F., Oppermann, T. and Müller, K., 2019. Automating the search for a patent's prior art with a full text similarity search. PLOS ONE, 14(3), p.e0212103.
[8] Li, S., Hu, J., Cui, Y. and Hu, J., 2018. DeepPatent: patent classification with convolutional neural networks and word embedding. Scientometrics, 117(2), pp.721-744.
[9] Lu, Y., Xiong, X., Zhang, W., Liu, J. and Zhao, R., 2020. Research on classification and similarity of patent citation based on deep learning. Scientometrics, 123(2), pp.813-839.
[10] Sarica, S., Song, B., Low, E. and Luo, J., 2019. Engineering Knowledge Graph for Keyword Discovery in Patent Search. Proceedings of the Design Society: International Conference on Engineering Design, 1(1), pp.2249-2258.
[11] PROJECT PQAI, 2020. Prior-Art Search for Everyone. https://projectpq.ai/
[12] Guo, J., Fan, Y., Ji, X. and Cheng, X., 2019. MatchZoo. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.
[13] HUGGING FACE, 2021. The AI community building the future. https://huggingface.co/
[14] EXPLOSION, 2021. Industrial-Strength Natural Language Processing. https://spacy.io/
[15] APACHE FOUNDATION, 2013. Apache UIMA. https://uima.apache.org/
[16] Klenner, A., Bergmann, S., Zimmermann, M. and Romberg, M., 2012. Large scale chemical patent mining with UIMA and UNICORE. Journal of Cheminformatics, 4(S1).
[17] Krishna, A., Feldman, B., Wolf. J, Gabel, G., Beliveau, S., Beach, T., 2016. Examiner Assisted Automated Patents Search. The 2016 AAAI Fall Symposium Series, Technical Report FS-16-02