Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track - Short paper

Combining Logic and Natural Language Processing to Support Investment Management

Marjolein Deryck[1,2] , **Nuno Comenda**[3] , **Bart Coppens**[3] , **Joost Vennekens**[1,2]
[1]KU Leuven, Dept. Computer Science, Campus De Nayer
[2]Leuven.AI – KU Leuven Institute for AI, Leuven, Belgium
[3]Coppens and Partners Consulting
{marjolein.deryck, joost.vennekens}@kuleuven.be,
{nuno.comenda,bart.jan.coppens}@coppens-and-partners.com

Abstract

This paper presents an application that we developed to assist users with the creation of an investment profile for the selection of financial assets. It consists of a natural language interface, an automatic translation to a declarative FO(.) knowledge base, and the IDP reasoning engine with multiple forms of logical inference. The application speeds up the investment profile creation process, and reduces the considerable inherent operational risk linked to the creation of investment profiles.

1 Introduction

The Knowledge Base Paradigm (KBP) advocates a strict separation between declarative domain knowledge and logical inference tasks that can be applied to this knowledge to solve problems of interest (Denecker 2008). As an implementation of this paradigm, the IDP system uses the expressive extension FO(.) of classical first order logic as a language for constructing a knowledge base (KB). The (.) stands for the various extensions, such as arithmetic, types, partial functions and inductive definitions (De Cat et al. 2018). In this application we especially use arithmetic and definitions. IDP offers an imperative shell for applying various logical inference algorithms to the KB. In this paper, we present a system that integrates an IDP application with a Natural Language (NL) interface to easily create the KB.

Multiple real-life applications have been created with IDP, e.g., for product configuration (Aerts and Vennekens 2018) and the calculation of registration duties (Deryck et al. 2019). Other authors have studied the use of NL to create a KB (Hoherchak, Darchuk, and Kryvyi 2021),(Khanam, Liu, and Chen 2019). Our contribution in this paper is to combine the two in order to tackle a real life application in the financial sector.

In the next section we introduce the case. Then we focus on the two levels of the application: the natural language interface in Section 3 and the reasoning engine with its inference tasks in Section 4. In the final sections we evaluate the application (Section 5) and address the lessons learned (Section 6) and conclusion (Section 7).

2 Case: Creation Of Investment Profiles

In this paper we report the results from a case study executed at an international financial institution. Due to the confidentiality required by the institution, some details of the case have been changed without affecting its essence.

As a part of its service, an investment banker offers clients advice on the financial products to buy or sell. The clients' preferences can be expressed in an investment profile, that determines which assets are eligible for a specific investment. The eligibility of a specific asset depends on a plethora of interacting rules and constraints. The bank uses software to automatically select assets according to the client's selection criteria. Previously, a bank operator translated the several requests into lengthy programs that contain a lot of enumerations, repetitions, and complex nesting of if-then clauses and exceptions that need to be followed in the right order. This makes each creation of an investment selection program a complex and time consuming task. Furthermore, the result is hard to validate, which entails a substantial operational risk.

3 A Natural Language Interface

To overcome the aforementioned challenges we propose a declarative reasoning system with a NL interface. It allows the eligibility of financial products to be defined by means of controlled natural language (CNL). Each sentence is constructed from a number of building blocks that are selected step by step to get to a complete sentence (see Fig.1). The resulting highly structured NL sentence is automatically translated to FO(.).

The building blocks are defined in a manually created *tuple tree*. The standard tree in our application consists of 400 nodes and took two weeks to construct. Each node represents a concept, like country or asset type, and each concept accepts possible values related to it. Users can extend the available picklist of values by introducing and defining new concepts. E.g., in the standard tuple tree 'rating' can take different values, from AAA, AA, A,... to D. A user could create a definition 'Investment Grade', defining it as ratings above BBB. 'Investment Grade' then becomes a new value in the picklist, that can be used to construct CNL sentences.

In addition to the interface to construct CNL sentences, the application also contains a deep learning NLP module

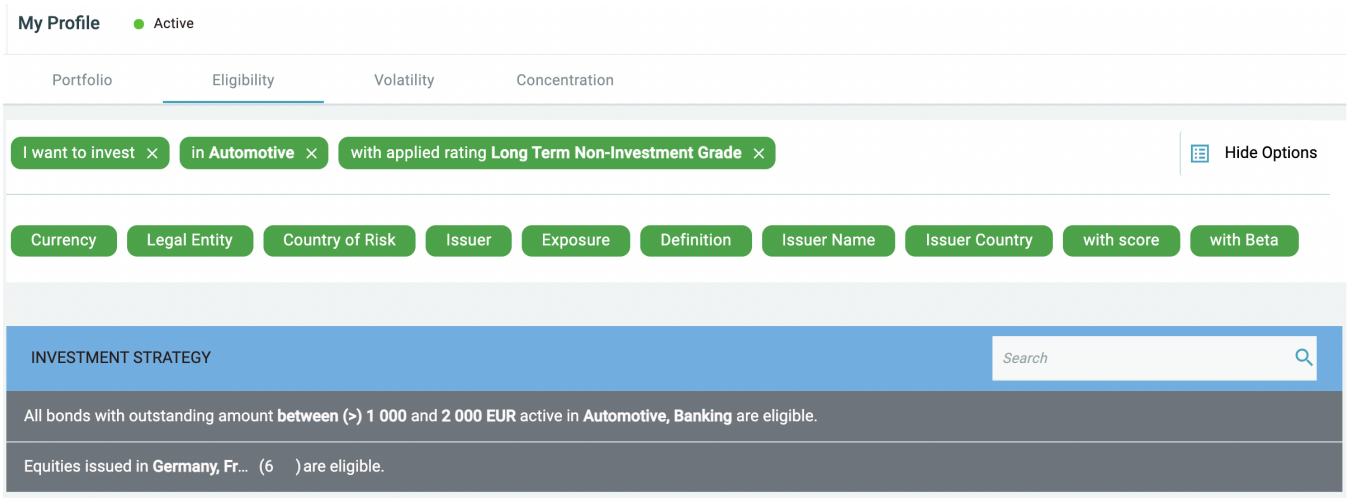

Figure 1: Creating a CNL sentence with building blocks.

that accepts free-form English. It proposes three CNL statements that are most likely to present the English sentence (see Fig.2). The user then selects the most correct sentence, makes adjustments if necessary, and validates the result. As before, this CNL statement is added to the KB in FO(.). The NLP module consists of a custom attention-based network, that was implemented in Tensorflow. The models use a sequence-to-sequence architecture with attention. The training data uses a combination of real and synthetic data. The real data are previous interactions from the users. The synthetic data is achieved by creating random walks on the tuple tree and several grammatical transformations to achieve a rich set of examples. The model is trained with an NVIDIA RTX 2080 with training times from tree to five hours.

The possibility to enter sentences in (controlled) natural language allows the users to create a complete and correct KB of their investment profile.

4 The Knowledge Base System

When completed, the KB can be used by different inference methods to perform multiple tasks in the problem domain. This section discusses the underlying IDP reasoning engine of the application, which consists of a KB at one hand, and inference tasks that can be applied to the KB on the other hand.

4.1 Structure Of The KB

A KB consists of three parts: a vocabulary that contains the ontology of the domain, a theory that contains rules and constraints on the concepts in the vocabulary, and a structure, that delineates the domain of the concepts, and typically gives an interpretation for some of them.

Deciding On Eligibility The KB is constructed by translating the CNL sentences to FO(.). The CNL sentences can end in one of two possible ways: either the asset is *eligible,* or it is *not eligible*. Every sentence is created independently of other sentences. This means that for a given asset, contradicting rules might occur. In this case, the non-eligibility rule should take precedence over the eligibility rule. To reflect this in our theory, we create two separate predicates *'Eligible(asset)'* and *'NotEligible(asset)'*. It is the predicate *'Eligible(asset)'* that determines if an asset is eligible for selection in the profile. The theory consists of two definitions, one for each predicate. A definition in FO(.) consists of a set of rules of the form $head \leftarrow body$. In our setting, all rules in the same definition have the same head. Each body is a sufficient condition for the general head to be true. Together, the rules are also necessary, i.e., the head can only be true if at least one body is (Denecker and Vennekens 2014). The *NotEligible(asset)* is used as an exception in the definition of *Eligible(asset)*. The general structure of our theory is shown in the example below:

$$\left\{ \begin{array}{l} \forall\, a[Asset] : Eligible(a) \leftarrow \varphi_1 \,\&\, \neg NotEligible(a). \\ \forall\, a[Asset] : Eligible(a) \leftarrow \varphi_2 \,\&\, \neg NotEligible(a). \\ \forall\, a[Asset] : Eligible(a) \leftarrow \varphi_3 \,\&\, \neg NotEligible(a). \end{array} \right\}$$

$$\{\ \forall\, a[Asset] : NotEligible(a) \leftarrow \varphi_4.\quad \}$$

These two definitions are equivalent to the formulas:

$$\forall\, a[Asset] : NotEligible(a) \Leftrightarrow \varphi_4.$$

$$\forall\, a[Asset] : Eligible(a) \Leftrightarrow \neg NotEligible(a) \wedge (\varphi_1 \vee \varphi_2 \ \vee\ \varphi_3).$$

Maintaining Concentration Limits Typically, an investor will not only describe the kind of assets they want to accept in their investment portfolio, but they will also have requirements for the profile of his entire portfolio. A sound portfolio should be diverse (e.g., in terms of asset countries or industries) and investors implement this by defining *concentration limits*. For the enforcement of these concentration limits we create a separate theory, as it concerns the next step after determining the eligibility, i.e., the actual selection of eligible assets. An example to force concentration limits on countries is shown below. We start by

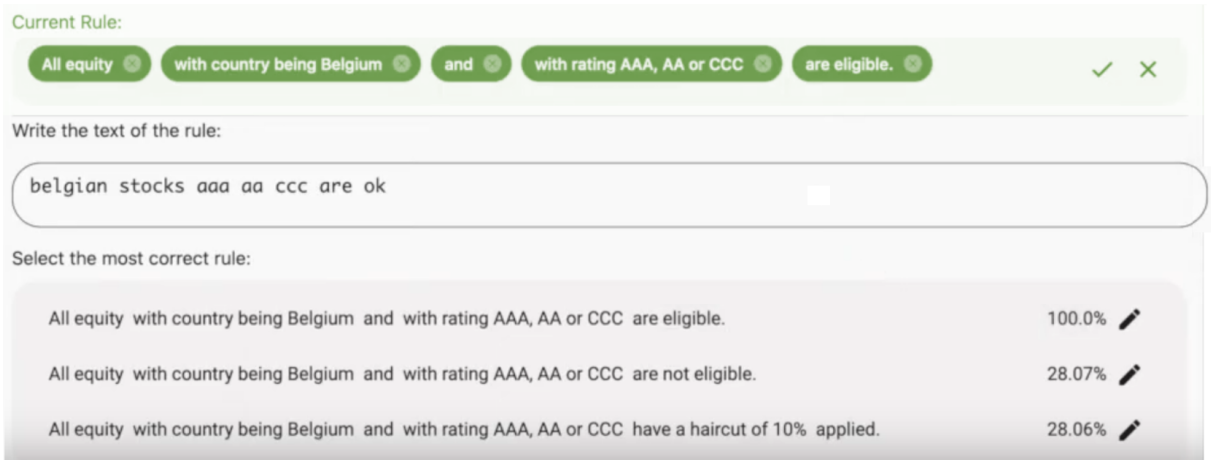

Figure 2: Natural language statement and its interpretations.

calculating the concentration amount of each country, that is the sum of the market values of each asset with the same country:

$$\forall c: \; CA(c) = sum\{a\ v : Country(a) = c \wedge MV(a) = v : v\}. \quad (1)$$

with *CA(c)* the concentration amount of country c, c the *Country* of asset a and v the market value *(MV)* of asset a. The sum-aggregate consists of three components : the variables a and v over which it ranges, a condition that a and v must satisfy ($Country(a) = c \wedge MV(a) = v$), and the value that needs to be summed up (v). The concentration, expressed as percentage, is given by

$$\forall\, c: \; CONC(c) \; = \; (CA(c) \times 100/TMV). \quad (2)$$

with *CONC(c)* the concentration percentage of country *c* and *TMV* the total market value of the portfolio.

Finally we force the concentration limit by comparing the user-set concentration limit percentage l of each country c (*CL(c,l)*) with the concentration percentage in the portfolio:

$$\forall\, c\, l: \; CL(c,l) \geq \; CON(c)\,. \quad (3)$$

4.2 Inference Tasks

The information that is declaratively stated in the KB, can be used for different purposes. This section describes how different inference tasks can be applied to implement different requirements. Several of these inference tasks take as input not just the KB T itself, but also a partial interpretation I_p for part of the vocabulary V of T.

Model Expansion The inference task of $model\ expansion$ can be used to decide on the eligibility of a specific asset. Given a theory T (that contains the rules of eligibility), and an interpretation I_p for part of its vocabulary V, the $model\ expansion$ inference computes interpretations I_t for the entire V such that $I_p \subset I_t$ and $I_t \models T$ (Wittocx, Mariën, and Denecker 2008). Consider the following highly simplified theory. If an asset has a triple A rating, it is investment grade (IG). If the asset is IG and is issued in Belgium (BE), it is eligible:

$\{\forall\, a[Asset] : IG(a) \leftarrow Rating(a) = AAA.\}$
$\{\forall\, a[Asset] : Eligible(a) \leftarrow IG(a) \wedge Country(a) = BE.\}$

The interpretation I_p that contains $Rating(asset) = AAA$, has two model expansions I_t:

$Rating(asset) = AAA$	$Rating(asset) = AAA$
$IG(asset) = true$	$IG(asset) = true$
$Country(asset) = BE$	$Country(asset) = FR$
$Eligible(asset) = true$	$Eligible(asset) = false$

In our application we typically possess all the information on the asset, such that only the values of *Eligible* and *NotEligible* need to be computed.

Optimization The $optimize$ inference is used to find a combination of eligible assets that can be acquired at minimal cost. To this end we create an additional term m that represents this cost. Given a theory T, interpretation I_p and term m, the $optimize$ inference will look for a model expansion I_t of I_p that minimizes m (De Cat et al. 2018). This is, it will select a combination of assets with the lowest associated cost that follows the eligibility rules and given I_p.

Propagation The $propagation$ inference computes a set of facts that are consequences of T given I_p, i.e., that hold in all model expansions I_t of T with $I_p \subset I_t$ (De Cat et al. 2018). In the example above, if the partial interpretation I_p contains $Rating(asset) = AAA$, than the value $IG(asset) = true$ is propagated, since it will be true in every possible model. Nothing can be derived about the value of $Eligible(asset)$ because this still depends on the country. In the application, the propagation works interactively: as soon as a new rule is created, the impact on the eligibility is immediately shown by coloring the asset green (eligible) or red (not eligible) as shown in Fig.3.

Explanation The $explanation$ inference traces the propagated values back to the given values of the interpretation

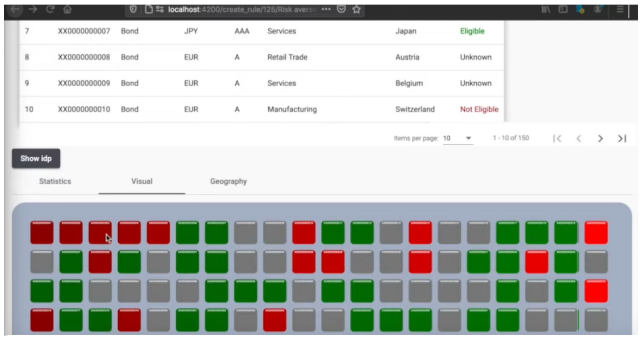

Figure 3: Visualization of interactive propagation.

I_p (Deryck et al. 2019). The application allows the user to click on a propagated value and see immediately which atoms steered the decision. This helps the user to understand unexpected propagations, for example in case an asset is not eligible, when the client expected that it would be. In the running example, the explanation of $IG(asset) = true$ is that $Rating(asset) = AAA$.

Explain Unsat In Section 4 we described the theory to reflect preferences with regards to concentration limits. When formalized this way, an existing profile can be checked to see if it satisfies the given concentration limits with a simple model expansion. If one of the limits is breached, no model will be found, and the $explain\ unsat$ inference task can be used to find which limit is breached.

Theory Comparison Active investors will typically update their profile regularly. In this case an automated comparison of two versions of the profile is helpful to ensure that correct amendments have been made. With the $model\ expansion$ inference the logical equivalence of two theories can be checked by merging two theories T_1 and T_2, with concepts $Eligible1$ and $Eligible2$ respectively, such that $T_3 = T_1 \cup T_2$ and adding the constraint that an asset can only be eligible in one of both theories. If no model I_t that satisfies T_3 is found, the two theories are equivalent. If differences are found, the model expansion inference will return the model that contains these differences, which allows their manual validation.

5 Evaluation

As the saying goes: the proof of the pudding is in the eating. The inference tasks were showcased to the company in a prototype. Following this, the company has launched a project to further develop this prototype into a production application. The first technical release in production was done in February 2021 and a second release with improved workflow for signing the profiles between counterparties was released in June 2021. As of the second release, clients from large investment banks have access to a sandbox environment for training purposes. A full commercial roll-out will be done by September 2021. The target users for this commercial release are operations teams in the treasury back offices of large investment banks globally (target around 500 users across 150 organisations).The correctness of the knowledge base was insured by performing empirical tests. The testing of eligibility of assets against the profile has been done for descriptions with up to 20 rules, applied to portfolios of up to 300 assets with response times less than 3 seconds. These represent reasonable tranches for proper business use. Any larger portfolios can be tested off line with reporting being sent when processing has finished.

Profiles of 20 lines in NL might take up to 2000 to 3000 ”if-then-else” statements when entered into the legacy systems. This is due to the large number of attributes (up to 150), the long lists of values behind most of these attributes (asset taxonomies, lists of countries, currencies, industries, ratings, issuers, etc.) and many overlaps and gaps in terms of eligibility. The ”if-then-else” approach of the legacy system forces a sequential way of organising the information in the profiles, leading to thousands of statements. Our FO-based approach allows for an almost line-by-line translation of the statements into logic. This leads to increased self-service by end clients, less operational errors and facilitated maintenance.

6 Lessons Learned

Previous case studies have shown that the Knowledge Base Paradigm and its implementation in the IDP system is very useful to develop powerful applications in knowledge-intensive domains (Aerts and Vennekens 2018; Deryck et al. 2019). This also proves to be the case here. Having the domain knowledge separated from the inference tasks keeps the maintenance of the code easy and lean. This is a big advantage over the way this kind of applications are usually governed. The addition of new concepts with minimum effort as discussed in Section 3 is an example of this. Typically, the creation of a KB is a challenging task (Feigenbaum 1977). Here, our NL interface offers two main advantages. Because of the complexity of the domain, detecting and fixing errors in the implementations of the policies used to be a time and resource intensive process. Now, the interface allows clients to enter the rules of their profile themselves, and immediately see the impact of them. The use of building blocks to create CNL sentences also has the advantage that it helps to elicit user's preferences, that might have stayed hidden without prompting by the building blocks. For instance, the start of a CNL sentence shows among others the following blocks: all bonds, all equities, all cash,... While the use of cash as investment is often overlooked, this screen prompts the user to consider it.

7 Conclusion

The prototype that we have developed strongly facilitates the creation of investment profiles. Compared with the manual creation of such profile, the operational risk linked to the automation is almost non-existent thanks to the two-step procedure to turn natural language sentences via CNL automatically into an FO(.) KB. Once the KB is created, the application supports multiple services, such as the selection of eligible assets, optimisation of the associated costs and explanation of unexpected results.

Acknowledgements

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

References

Aerts, B., and Vennekens, J. 2018. An application of logic-based methods to machine component design. volume 64, 13:1–13:15. Palù, Alessandro Dal.

De Cat, B.; Bogaerts, B.; Bruynooghe, M.; Janssens, G.; and Denecker, M. 2018. Predicate logic as a modeling language: The idp system. In *Declarative Logic Programming: Theory, Systems, and Applications*. ACM Books. 279–329.

Denecker, M., and Vennekens, J. 2014. The well-founded semantics is the principle of inductive definition, revisited. 1–10. Chitta, Baral.

Denecker, M. 2008. Building a knowledge base system for an integration of logic programming and classical logic. volume 5366, 71–76. Springer.

Deryck, M.; Devriendt, J.; Marynissen, S.; and Vennekens, J. 2019. Legislation in the knowledge base paradigm: interactive decision enactment for registration duties. 174–177. IEEE.

Feigenbaum, E. A. 1977. The art of artificial intelligence. 1. themes and case studies of knowledge engineering. Technical report, Stanford Univ CA Dept of Computer Science.

Hoherchak, H.; Darchuk, N.; and Kryvyi, S. 2021. Representation, Analysis, and Extraction of Knowledge from Unstructured Natural Language Texts. *Cybernetics and Systems Analysis* 57(3):481–500.

Khanam, S. A.; Liu, F.; and Chen, Y.-P. P. 2019. Comprehensive structured knowledge base system construction with natural language presentation. *Human-centric computing and information sciences* 9(1):1–32.

Wittocx, J.; Mariën, M.; and Denecker, M. 2008. The idp system: a model expansion system for an extension of classical logic. In *Proceedings of the 2nd Workshop on Logic and Search*, 153–165. ACCO; Leuven.