

FRIEDRICH-ALEXANDER-UNIVERSITÄT
ERLANGEN-NÜRNBERG



PROFESSUR FÜR WISSENSREPRÄSENTATION UND -VERARBEITUNG
KWARC GROUP

A Symbolic Approach to Job Recommendation

Seminar Wissensrepräsentation und -verarbeitung

Zambou Zoleko Pascal Pierre

Advisors:

Prof. Dr. Michael Kohlhase



Erlangen, September 30, 2020

Abstract

In practice, employment advisors are not able to keep track of all the knowledge about occupations and their required skills. As a result, consultants often recommend trivial occupations based on a person's individual profile. However, in today's labour market (especially in times of crisis when many people lose their jobs) it can be difficult to find a similar job in exactly the same sector. Therefore, in this work, I present an algorithm based on symbolic techniques that can recommend trivial and non-trivial professions based on a person's profile. Once implemented, such a knowledge-based system can help to expand the relevant search space for a job applicant.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 2 | Knowledge Acquisition and Representation | 2 |
| 2.1 | Data Collection | 2 |
| 2.2 | Preprocessing | 2 |
| 2.3 | From Data to Knowledge Representation | 3 |
| 3 | Inference | 4 |
| 3.1 | Naive Inference | 4 |
| 3.2 | Fuzzy Inference | 5 |
| 4 | Evaluation | 8 |
| 5 | Future Work | 10 |
| 6 | Conclusion | 11 |
| 7 | Appendix | 12 |
| 7.1 | The ABox | 12 |
| 7.2 | The ABox in \mathcal{ALC} | 12 |
| 7.3 | Fuzzy Logic | 13 |
| | References | 15 |

1 Introduction

The transition from one job to another is not easy and often depends on many factors, such as required skills, location or salary [8]. Since a successful job transition is essential for a successful career [8], individuals sometimes turn to experts or employment consultants to help them make the right decision. However, there is a wide variety of professions and skills making it is very difficult for a consultant to keep track of them all. Consequently, this limitation of the human capabilities can lead to poor job recommendations. Although it could be argued that employment consultants today would use a computer to access the vast amount of knowledge surrounding the labour market, these systems are often used as simple (hit or miss) search engines that can at best, find trivial relationships between occupations and skills.

For example, suppose there is an event photographer who has just lost his job. Let us call this photographer John. John visits Mary, a personnel consultant, in the hope of finding a new job. Unfortunately for John, the current crisis makes it impossible for him to obtain a similar job in the same field. For the sake of example, let's further assume that Mary types in John's former profession (event photographer) into a database to recommend new opportunities and finds examples such as event manager, wedding photographer and landscape photographer. Standard database checks such as these are often based only on syntax. Even if the returned jobs are correct (in the sense that they are related to the previous profession), John cannot perform them due to the current crisis. I call these "trivial recommendations".

In this work, I present an algorithm that borrows some concepts from fuzzy description logic, especially fuzzy ALC and graph algorithms. The goal of the algorithm¹ is to find trivial and non-trivial career opportunities based on the profile of a person. Returning to our example, such an algorithm could help John to find career opportunities in social media management or web design, since similar skills like Adobe Photoshop and Illustrator are regularly used by professionals in these fields. As mentioned above, other factors are usually taken into account to make a good recommendation. To keep things simple, the proposed algorithm only takes as input a person's profile (i.e. a set of competencies and past professions) and then returns the top N jobs matching the profile.

In the next chapter, I briefly explain how knowledge is acquired and represented. Chapter 3 describes how inferences are made using the knowledge acquired. Evaluation is conducted in chapter 4. Future work is discussed in chapter 5, followed by the conclusion. Refer to the appendix for preliminaries.

¹This algorithm was implement into a job recommendation system available only under restricted access at <https://ontology-ui.greple.ai/>.

2 Knowledge Acquisition and Representation

In order to build a job recommendation system, one must first acquire knowledge. Data on the labour market, occupations and skills can be collected from various sources such as social media platforms, job portals, public databases and through web crawling, each with its advantages and disadvantages.

2.1 Data Collection

Social media platforms, for example, hold huge amounts of data which can be used to generate knowledge. However, this data is often unstructured and thus, a considerable amount of time has to be invested on data cleaning. In addition, social media data is subject to data protection laws like GDPR (DSGVO), which further complicate business processes. While sources such as job portals reflect the current job market, most of the available data is in English and skewed towards white-collar professions.

Fortunately, there are sources that provide structured data on jobs, skills and their relationships in German for both blue-collar and white-collar jobs. The source used in this work is available at www.ams.at. AMS contains qualitative and up-to-date information on the labour market. For each profession on AMS, there is information about the required skills and certifications, related professions and specialisations.

2.2 Preprocessing

Once the data is collected, minor pre-processing tasks are performed. For example, the German translations for photographer are “Fotografer” and “Fotografin”, one for each gender. So, to avoid ambiguities, the professions (and skills) are normalised by removing one of the two genders (masculine in this case). Once the data is ready, we can now represent it in a way that will facilitate its manipulation.

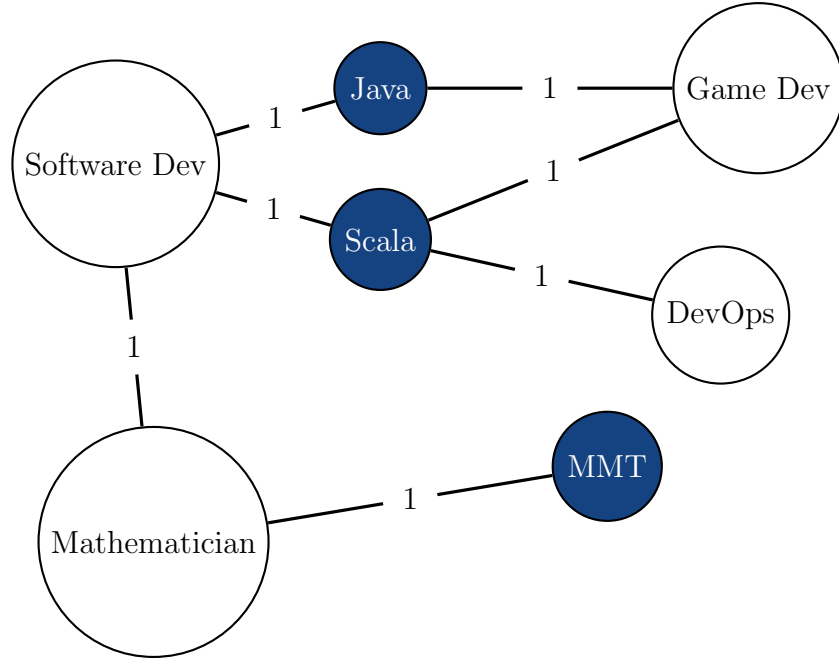


Figure 2.1: Depicting knowledge representation with graphs. Blue nodes represent skills and white nodes represent jobs

2.3 From Data to Knowledge Representation

One of the main advantages of AMS data is its structure. Each profession is associated to one or more skills. Related skills are also connected to each other. We can utilize this graph-like structure and explicitly create an undirected graph $G(E, V)$ from the data. The nodes V in G represent jobs and skills, while the edges E represent the relationship between the nodes. For simplicity, all edges are assumed to have a weight of 1.

At this point, the knowledge can be used to make inferences. This is where description logics (DLs), in particular \mathcal{ALC} comes in (see appendix). The next chapter discusses how \mathcal{ALC} is used at inference time.

3 Inference

Remember that the goal of this work is to recommend new career opportunities (or jobs) based on a person's profile.

Definition 1. *Let P denote a person's profile. P is a non-empty set consisting of jobs and skills. For example, $P = \{\text{MMT}, \text{Scala}\}$ is a profile consisting of two skills (see figure 2.1), namely MMT and Scala.*

To make inferences, the description logic language \mathcal{ALC} is used. \mathcal{ALC} can be used to make recommendations from the graph built in the previous section by constructing an ABox and then making inferences using the ABox. The following ABox is produced:

$$\begin{aligned} \mathcal{A}_{work} = \{ & \text{Java:Skill}, \text{MMT:Skill}, \text{Mathematician:Job}, \text{DevOps:Job}, \text{Game Dev:Job}, \\ & \text{Software Dev:Job}, (\text{Java}, \text{Software Dev}):related, (\text{Java}, \text{Game Dev}):related, \\ & (\text{Mathematician}, \text{Software Dev}):related, (\text{Scala}, \text{Game Dev}):related, (\text{Mathematician}, \\ & \text{MMT}):related, (\text{Scala}, \text{DevOps}):related, (\text{Scala}, \text{Software Dev}):related, (\text{Software Dev}, \\ & \text{Java}):related, (\text{Game Dev}, \text{Java}):related, (\text{Software Dev}, \text{Mathematician}):related, (\text{Game} \\ & \text{Dev}, \text{Scala}):related, (\text{MMT}, \text{Mathematician}):related, (\text{DevOps}, \text{Scala}):related, (\text{Software} \\ & \text{Dev}, \text{Scala}):related \} \end{aligned}$$

3.1 Naive Inference

Suppose we are provided with a profile $P = \{\text{MMT}\}$ and we need to recommend the best job that matches this profile. (Throughout this work, I will continue using the graph from Figure 3.1 unless otherwise stated.) In the \mathcal{ALC} , this query corresponds to finding the jobs, which satisfy this term $\exists related.Job(MMT)$. In other words, we have to find the jobs that are related to MMT. Let \mathcal{I} be our interpretation function and let $\Delta^{\mathcal{I}}$ be our domain. The term $\exists related.Job(MMT)$ can be interpreted as follows:

$$\exists related.Job^{\mathcal{I}}(MMT) = \{\exists j. (MMT, j) \in related^{\mathcal{I}} \cap j \in Job^{\mathcal{I}}\}$$

which is non-empty iff there is a job j which is related to MMT . There is only one job (*Mathematician*), for which the term is true. Although the prediction is correct, I call this approach naive, because it does not handle multiple inputs well and crisp predictions do not provide a satisfactory measure to compare predictions.

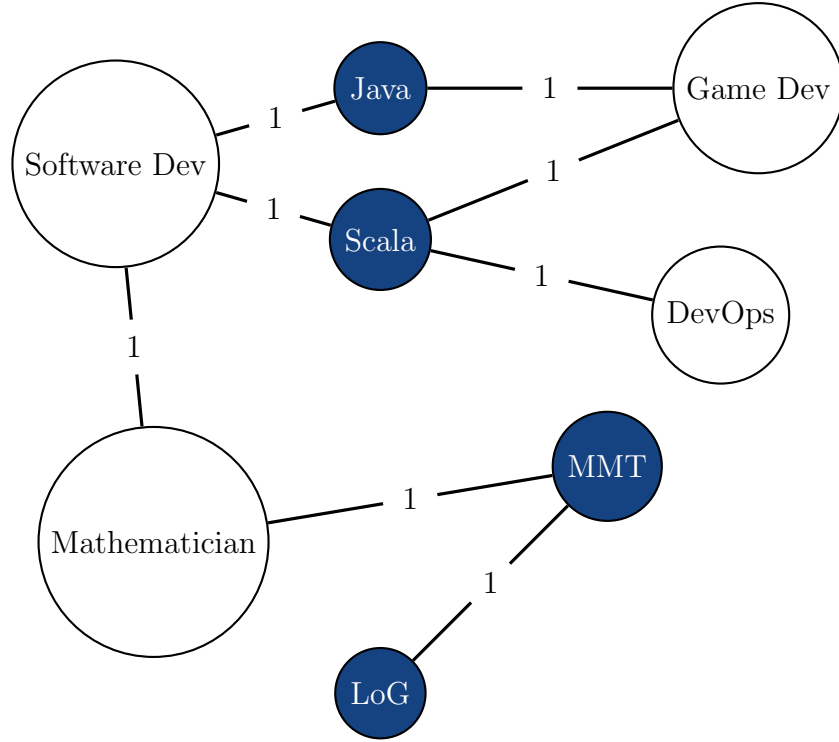


Figure 3.1: The previous graph is extended by adding a new skill called LoG

To better understand why, assume we are provided with a different profile this time: $P = \{\text{LoG}, \text{Scala}\}$. There are basically two ways one can make an inference given this profile.

- In the first case, we find the set of jobs that are related to both *LoG* and *Scala*, which in this case is just \emptyset or
- find the set of jobs related to *LoG* and the set of jobs related to *Scala* and then return their union. We would obtain $\emptyset \cup \{\text{SoftwareDev}, \text{GameDev}, \text{DevOps}\}$.

In the first situation, no job is recommended although the knowledge is not empty. In the second case, the skill *LoG* had no influence on the predictions. Either way, we have no way of ranking jobs effectively.

To overcome this challenge, \mathcal{ALC} is extended to support fuzzy truth values.

3.2 Fuzzy Inference

Unlike in the crisp \mathcal{ALC} , where we try to find jobs that satisfy the term $\exists \text{related.job}(\text{LoG}, \text{Scala})$, here we look for the jobs that maximize the truth value of the term. In other words, we need to find the jobs that *most likely* match the input profile. This calls for the following interpretation:

| | | | | | |
|-------|-----|--------------|---------------|----------|--------|
| | ... | Software Dev | Mathematician | Game Dev | DevOps |
| Java | | 1 | 2 | 1 | 3 |
| Scala | | 1 | 2 | 1 | 1 |
| LoG | | 3 | 2 | 5 | 5 |
| MMT | | 2 | 1 | 4 | 4 |
| ... | ... | ... | ... | ... | ... |

Table 1: All pairs shortest path matrix S.

$$\begin{aligned}
& (\exists \text{related.Job(LoG)} \sqcap \exists \text{related.Job(Scala)})^{\mathcal{I}} \\
& = \{ \sup_{j \in \Delta^{\mathcal{I}}} | \text{related}^{\mathcal{I}}(\text{LoG}, j) \otimes \text{related}^{\mathcal{I}}(\text{Scala}, j) \otimes \text{Job}^{\mathcal{I}}(j) \}
\end{aligned}$$

The combination function (see appendix) \otimes here is the equivalent of \sqcap in crisp \mathcal{ALC} . Relationships between skills and jobs no longer have a crisp interpretation ($\{0, 1\}$). Instead, the relations $\text{related}^{\mathcal{I}}(\text{LoG}, j)$ and $\text{related}^{\mathcal{I}}(\text{Scala}, j)$ return values in $[0, 1]$. For practical reasons, the concept assertion $\text{Job}^{\mathcal{I}}(j)$ is dropped because j is *chosen* iff it is in $\text{Job}^{\mathcal{I}}$. Therefore, $\text{Job}^{\mathcal{I}}(j)$ is always a constant ($= 1$) and can be ignored during computation. The combination function of choice for this work was the *t-norm* (see appendix) from *Product Logic* because it delivered the best results in practice. Comparable results were also obtained by using the *s-norm* but inference was very slow and thus impractical in a productive environment.

Example 1. Let $\text{related}^{\mathcal{I}}(\text{LoG}, \text{DevOps}) = 0.2$, $\text{related}^{\mathcal{I}}(\text{Scala}, \text{DevOps}) = 1.0$ then

$$\begin{aligned}
& \text{related}^{\mathcal{I}}(\text{LoG}, \text{DevOps}) \otimes \text{related}^{\mathcal{I}}(\text{Scala}, \text{DevOps}) \\
& = \text{related}^{\mathcal{I}}(\text{LoG}, \text{DevOps}) * \text{related}^{\mathcal{I}}(\text{Scala}, \text{DevOps}) = 0.2
\end{aligned}$$

where \otimes is the t-norm combination function from Product Logic.

So far it was convenient to assume that some interpretation function maps concept and role assertions to fuzzy values. Here are the steps needed to actually build such a function:

Consider the graph G in Figure 3.1

1. Calculate the shortest paths S between all node pairs in G [5], where S is a matrix and S_{ij} is the distance between all nodes i and j in G (see Table 1).
2. Transform S_i so that every S_{ij} in S_i can be within the range $[0, 1]$. For transformation, *minmax scaling* [1] is used (see Table 2).

$$S_{ij} = \frac{S_{ij} - \min(S_i)}{\max(S_i) - \min(S_i)}$$

3. Flip every S_{ij} in S so that the minimum becomes the maximum and vice versa.

$$S_{ij} = 1 - S_{ij}$$

| | | | | | |
|-------|-----|--------------|---------------|----------|--------|
| | ... | Software Dev | Mathematician | Game Dev | DevOps |
| Java | | 0 | 0.5 | 0 | 1 |
| Scala | | 0 | 1 | 0 | 0 |
| LoG | | 0.5 | 0 | 1 | 1 |
| MMT | | 0.33 | 0 | 1 | 1 |
| ... | ... | ... | | ... | ... |

Table 2: Scaled distance matrix S.

| | | | | | |
|-------|-----|--------------|---------------|----------|--------|
| | ... | Software Dev | Mathematician | Game Dev | DevOps |
| Java | | 1 | 0.5 | 1 | 0 |
| Scala | | 1 | 0 | 1 | 1 |
| LoG | | 0.5 | 1 | 0 | 0 |
| MMT | | 0.67 | 1 | 0 | 0 |
| ... | ... | ... | | ... | ... |

Table 3: Distance matrix S scaled and flipped.

In the first step, compute a measure of the proximity between the nodes. S_{ij} is a distance relation between two nodes i and j . Although S gives us a relations between nodes, its values are in the range of $[0, \infty]$ and therefore cannot be used as fuzzy truth degrees (since they must be in $[0, 1]$). This is why S_i is scaled in step two. After scaling, S_i now lies in the range $[0, 1]$. The last step ensures that $S_{ij} > S_{ij+1}$ iff j is closer to i than $j+1$. S is our interpretation function¹, mapping every relation $R(u, v)$ to $[0, 1]$.

Example 2. Which job should we recommend given the profile $P = \{\text{LoG}, \text{Scala}\}$?

Solution: Find the job j that maximises

$$R(j) = \text{related}^{\mathcal{I}}(\text{LoG}, j) \otimes \text{related}^{\mathcal{I}}(\text{Scala}, j).$$

$$R(\text{DevOps}) = 0 * 1 = 0$$

$$R(\text{SoftwareDev}) = 0.5 * 1 = 0.5$$

$$R(\text{Mathematician}) = 1 * 0 = 0$$

$$R(\text{GameDev}) = 0 * 1 = 0$$

$R(j)$ is maximised when $j = \text{Software Dev}$. Hence, *Software Dev* is returned as the top recommendation for the profile $P = \{\text{LoG}, \text{Scala}\}$.

¹Note that S is just a submatrix of some matrix S' , the real interpretation function that also maps concept assertions $C(u)^{\mathcal{I}}$ to $[0, 1]$. As discussed earlier, since we always choose u such that $C(u)^{\mathcal{I}} = 1$, this constant value is ignored during computation.

4 Evaluation

The proposed method showed great improvements in comparison to techniques previously tested at <https://ontology-ui.greple.ai/> (e.g Graph Learning). It is to be noted that the limited access to the system considerably hinders the evaluation process. Notwithstanding, evaluating such a system is a difficult task as predictions considered correct are highly subjective. Recall the example of John and Mary from the Introduction. The recommendations made by Mary were very bad given economic situation (in the fictive scenario), albeit rather correct and highly accurate in general.

The table below shows some predictions made by the system. The target recommendation is obtained by intersecting the jobs related to every skill or profession in the profile¹

The (few) test results in the table were selected at random. The proposed approach has an accuracy of 40%, $2 \times$ more than statistical approach previously investigated. Accuracy here is based on exact matching with the target job. However, predicted jobs that do not match with the target are still very similar e.g Elementarpädagoge/-pädagogin and KinderbetreuerIn. Conducting a reasonable evaluation is hence very difficult due to the degree of subjectivity involved in this task.

¹Sometimes there might be more than one job in common and so one is picked at random.

| Profile | Top Prediction: Fuzzy ALC | Actual Target on AMS | Top Prediction: Graph Learning TransE2 |
|--|---|-----------------------------|---|
| 3D-Design, Grafikerin | Web-Designer/-in | Web-DesignerIn | Computergrafiker/-in |
| Apache-Server, Betriebssystem- Configuration-Management, IT-Consultant (m/w/d) | Software-Entwickler/-in | InformatikerIn | IT-Systemspezialist/-in |
| Interkulturelles Lernen, Frühkindliche Erziehung, Lernfördernde Freizeitgestaltung | Elementarpädagoge/ -pädagogin | KinderbetreuerIn | Elementarpädagoge/ -pädagogin |
| AllgemeineR HilfsarbeiterIn, ExpeditarbeiterIn, GemeindearbeiterIn, Hilfskraft der Holzverarbeitung (m/w), LagerarbeiterIn, Landwirtschaftliche Hilfskraft (m/w), PlatzmeisterIn, Staplerschein | Produktionshilfskraft (m/w/d) | Produktionshilfskraft (m/w) | Hubstaplerfahrer/-in |
| Geige spielen, Gesang, Künstlerische Fachkenntnisse | Musiker/-in | Musiker/-in | Musiker/-in |
| Berechnung von Energiekennzahlen, Technische Bauplanung | Architekt/-in, Bautechniker/-in | EnergieberaterIn | Bautechniker/-in |
| Erstellen von Musikvideos, Erstellen von Unterrichtsfilmen, Film und Fernsehen, Videotechnik | Kameramann/-frau, Medientechniker/-in | Kameramann/-frau | Cutter/-in |
| ReisebüroassistentIn, ReiseleiterIn, TourismusmanagerIn | Hotelkaufmann/-frau | AnimateurIn | Rezeptionist/-in |
| Finanzsektor, Aktienhandel, Bank- und Finanzwesen-Kenntnisse | Bankkaufmann/-frau, Bankangestellte/-r, Finanz- und Anlageberater/-in | Börsenhändler/-in | Börsenhändler/-in |

Table 4: Predictions made by the proposed algorithm compared to the TransE model [4] trained on AMS data.

5 Future Work

The biggest limitation of the proposed approach is the memory footprint. A graph with ~ 70000 nodes and a 70000×70000 distance matrix is used during inference, requiring at least 4 GB of random access memory. Initially, the computing time grew exponentially with the increase in concurrent requests, which subsequently led to out-of-memory scenarios. This problem was solved by using Map-Reduce to compute the products. It is easy to see that the proposed approach is impractical for large graphs with millions of nodes. If the large distance matrix is replaced by a trainable function (like a neural network) that calculates the similarity between two nodes, the problem is solved at the expense of loss of accuracy. Further research must be done to confirm this claim. At last, there is a need to search for better evaluation metrics in highly subjective use cases like this one. Because of the limited access granted by the owner, only a few profiles were tested. Proper evaluation still has to be done more data.

6 Conclusion

In this paper a symbolic approach to recommending jobs was proposed. A detailed explanation of the proposed algorithm was also presented. Despite the limited access to the system implementing the algorithm, some of the results (predictions) were shared and compared with those predicted by a statistical model. The limitations of this work were discussed, and a hypothesis was put forward to overcome them. In conclusion, this work shows that despite the ongoing hype surrounding statistical methods, symbolic techniques can still prove to be quite effective.

7 Appendix

Description logics (DLs) are a family of knowledge representation languages used to describe and reason about knowledge in a structured and well-understood way [3]. DLs are constructed from two disjoint sets of symbols, namely atomic concepts and atomic roles [7]. Constructors can be used to build complex terms [7]. For instance,

$$\text{Graduate} \sqcap \text{Teacher}$$

denotes the set of individuals objects that are graduates and teachers.

A DL language can be divided into the *TBox* and the *ABox* [3]. The ABox is described briefly in the following subsection.

7.1 The ABox

In a nutshell, an ABox contains assertions on individuals [7]. For example, the *concept assertion*

$$\text{John:Photographer}$$

states that the individual object named John is a Photographer. Next to concept assertions, we have *role assertions*. Here is an example:

$$(\text{John}, \text{Adobe Photoshop}):uses$$

stating that John uses Adobe Photoshop.

7.2 The ABox in \mathcal{ALC}

The Attributive Language with Complements (\mathcal{ALC}) [9] uses an *interpretation* (\mathcal{I}) to determine the meaning of concept and role assertions. \mathcal{I} consists of a non-empty set of individuals called the *interpretation domain* ($\Delta^{\mathcal{I}}$) and an *interpretation function* that can tell us which elements in the domain belongs to a concept, or which pairs of elements are related to each other by a role [3].

As an example, consider the following ABox \mathcal{A}_{work} :

$$\mathcal{A}_{work} = \{ \text{John:Person}, \text{Photoshop:Skill}, (\text{John}, \text{Photoshop}):uses \}$$

where John:Person is an \mathcal{ALC} concept assertion and $(\text{John}, \text{Photoshop}):uses$ is a role assertion. An interpretation \mathcal{I} of \mathcal{A}_{work} can be defined as follows:

$$\begin{aligned} \Delta^{\mathcal{I}} &= \{u, v\}, \\ \text{John}^{\mathcal{I}} &= u, \\ \text{Photoshop}^{\mathcal{I}} &= v, \\ \text{Person}^{\mathcal{I}} &= \{u\}, \\ \text{Skill}^{\mathcal{I}} &= \{v\}, \\ \text{uses}^{\mathcal{I}} &= \{(u, v)\}. \end{aligned}$$

John is interpreted as u and the relationship *uses* between John and Photoshop is interpreted as (u, v) . Interpretation in \mathcal{ALC} is crisp, meaning an individual is either a member of a concept or not (True or False). This crisp interpretation is not very useful when comparing individuals. For example, the role assertion $(\text{John}, \text{Photoshop}):uses$ does not express to what degree John uses Photoshop or how likely John is to use Photoshop. To overcome this limitation in \mathcal{ALC} , the language is extended so that it can handle fuzzy truth values [10].

7.3 Fuzzy Logic

Suppose John is a photographer and Mary a Software Developer. To work, they both have to use Photoshop from time to time. We can create a binary predicate *Uses* with $\text{Uses}(\text{John}, \text{Photoshop})$ and $\text{Uses}(\text{Mary}, \text{Photoshop})$ set to **True**. The predicate *Uses* however, does not tell us how much Photoshop is used by John and Mary, or how likely is it to be used by them. In fact, there is no way to compare these relations when they have the same truth value.

Fuzzy logics enable us to define a function that maps propositions or predicates to a range of truth values [6]. In fuzzy logic therefore, an element belongs to a set to a certain degree n in $[0, 1]$, where n is called the *degree of truth* of a proposition ϕ in an interpretation \mathcal{I} [6].

In First Order Logic for instance, a fuzzy interpretation \mathcal{I} maps atomic propositions (and predicates) φ_i into $[0, 1]$. Below are some examples showing how \mathcal{I} is extended using combination functions [11]:

$$\begin{aligned} \mathcal{I}(\phi \wedge \psi) &= \mathcal{I}(\phi) \otimes \mathcal{I}(\psi) \\ \mathcal{I}(\phi \vee \psi) &= \mathcal{I}(\phi) \oplus \mathcal{I}(\psi) \\ \mathcal{I}(\exists x. \phi) &= \sup_{a \in \Delta^{\mathcal{I}}} \mathcal{I}_x^a(\phi) \end{aligned}$$

where

- \otimes and \oplus are called *combination functions*, (*t-norms* and *s-norms* respectively), which extend the classical Boolean conjunction and disjunction to fuzzy truth degrees [6].
- *Supremum* (**sup**) is used to model the existential quantifier [2].

There are different types of fuzzy logics, e.g. *Gödel*, *Lukasiewicz* and *Product logic* [11] which all define combination function differently. A comparison between Łukasiewicz and Product Logic found in Table 5.

| | Łukasiewicz Logic | Product Logic |
|---------------|----------------------|---------------------|
| $a \otimes b$ | $\max(a + b - 1, 0)$ | $a \cdot b$ |
| $a \oplus b$ | $\min(a + b, 1)$ | $a + b - a \cdot b$ |

Table 5: Combination functions in Łukasiewicz and Product Logic [6].

Back to our example, if $\text{Uses}(\text{John}, \text{Photoshop}) > \text{Uses}(\text{Mary}, \text{Photoshop})$, this could mean that John uses Photoshop more than Mary for his work, or that he is more likely to use Photoshop.

References

- [1] Amanda Casari Alice Zheng. *Feature Engineering for Machine Learning*. O'Reilly Media, Inc., 2018.
- [2] F. Baader, R. Küsters, and F. Wolter. *Extensions to Description Logics*, page 237–282. Cambridge University Press, 2 edition, 2007.
- [3] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *Introduction*, page 1–9. Cambridge University Press, 2017.
- [4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.
- [5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.
- [6] Thomas Lukasiewicz and Umberto Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *J. Web Sem.*, 6(4):291–308, 2008.
- [7] D. Nardi and R. J. Brachman. *An Introduction to Description Logics*, page 1–44. Cambridge University Press, 2 edition, 2007.
- [8] Ioannis K. Paparrizos, Berkant Barla Cambazoglu, and Aristides Gionis. Machine learned job recommendation. In *Proceedings of the Fifth Conference on Recommender Systems*, pages 325–328. ACM, 2011.
- [9] Sebastian Rudolph. Foundations of description logics. In *Reasoning Web*, volume 6848 of *Lecture Notes in Computer Science*, pages 76–136. Springer, 2011.
- [10] Lutz Schröder and Dirk Pattinson. Description logics and fuzzy probability. In *IJCAI*, pages 1075–1081. IJCAI/AAAI, 2011.
- [11] Umberto Straccia. All about fuzzy description logics and applications. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*, pages 1–31, 2015.

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.