Automated Rule Base Completion as Bayesian Concept Induction

Zied Bouraoui
CRIL - CNRS & Univ Artois, France
zied.bouraoui@cril.fr

Steven Schockaert
Cardiff University, UK
SchockaertS1@Cardiff.ac.uk

Abstract

Considerable attention has recently been devoted to the problem of automatically extending knowledge bases by applying some form of inductive reasoning. While the vast majority of existing work is centred around so-called knowledge graphs, in this paper we consider a setting where the input consists of a set of (existential) rules. To this end, we exploit a vector space representation of the considered concepts, which is partly induced from the rule base itself and partly from a pre-trained word embedding. Inspired by recent approaches to concept induction, we then model rule templates in this vector space embedding using Gaussian distributions. Unlike many existing approaches, we learn rules by directly exploiting regularities in the given rule base, and do not require that a database with concept and relation instances is given. As a result, our method can be applied to a wide variety of ontologies. We present experimental results that demonstrate the effectiveness of our method.

1 Introduction

The problem of automated knowledge base completion has received considerable attention in recent years (Pujara *et al.* 2017). Within the broad aim of knowledge base completion, various strategies can be explored. One possible strategy is to find missing facts by searching for relevant documents on the Web, and analyzing their content (West *et al.* 2014). Another strategy is to identify and exploit statistical regularities among the facts in a given knowledge base (Lao *et al.* 2011; Bordes *et al.* 2013). Most existing approaches, however, focus on finding plausible missing *facts*. Our focus in this paper is instead to find missing knowledge in a given ontology.

In particular, we propose an approach to find plausible rules which is inspired by cognitive models for category based induction (Osherson *et al.* 1990; Tenenbaum and Griffiths 2001). The main aim of such induction models is to determine which objects are likely to have some property P, knowing that the objects $o_1, ..., o_n$ have this property (but knowing nothing else about property P). In other words, inductive generalization in these models is based on our knowledge of the semantic features of the objects. For example, knowing that oranges, lemons and grapefruit have some unknown property P, we can plausibly derive that limes have this property as well, simply because there are very few natural properties which hold for oranges, lemons and grapefruit, but not for lime. Similarly, suppose we have a knowledge base containing rules of the following form:

$$r_1(X) \wedge \textit{orange}(X) \rightarrow r_2(X)$$
$$r_1(X) \wedge \textit{lemon}(X) \rightarrow r_2(X)$$
$$r_1(X) \wedge \textit{grapefruit}(X) \rightarrow r_2(X)$$

Without knowing anything about the meaning of the relations r_1 and r_2, we can intuitively still derive that the following rule is plausible:

$$r_1(X) \wedge \textit{lime}(X) \rightarrow r_2(X)$$

To implement this intuition, we rely on two types of vector space representations of the considered relations. First, we can use word embeddings (Mikolov *et al.* 2013; Pennington *et al.* 2014), which are vector space representations of word meaning that are learned from large text collections. Such representations have been found to exhibit various interesting regularities, which means that they can be regarded as a source of commonsense knowledge (Levy *et al.* 2014; Gupta *et al.* 2015). Most importantly for our purposes, words representing concepts with similar properties, such as different kinds of citrus fruits, tend to be clustered together. Second, we will also rely on a vector space representation that has been learned from the ontology itself. The aim of this representation is to capture the intuition that relations which are asserted to have similar properties should also be considered as similar for the purpose of inductive reasoning.

Similarly, by focusing on pairs of concepts, we can also complete rule bases using a form of analogical reasoning. Consider the following example:

$$r_1(X, Y) \wedge \textit{bat}(X) \rightarrow \textit{cave}(Y)$$
$$r_1(X, Y) \wedge \textit{duck}(X) \rightarrow \textit{pond}(Y)$$
$$r_1(X, Y) \wedge \textit{dolphin}(X) \rightarrow \textit{sea}(Y)$$

Then we can plausible also derive the following rule, based on the analogical relationship that holds between the pairs $(\textit{bat}, \textit{cave})$, $(\textit{duck}, \textit{pond})$, $(\textit{dolphin}, \textit{sea})$ and $(\textit{trout}, \textit{river})$:

$$r_1(X, Y) \wedge \textit{trout}(X) \rightarrow \textit{river}(Y)$$

Such analogical relationships can again be effectively identified from word embeddings (Mikolov *et al.* 2013), and other types of vector space representations.

To implement the aforementioned ideas for completing sets of rules, we will consider rule *templates*, such as

$$\tau(\star) = r_1(X) \wedge \star(X) \rightarrow r_2(X)$$

These templates are second-order relations, whose instances are the relations from the ontology. For example, the concepts *orange*, *lemon* and *grapefruit* would be instances of the template τ. This view will allow us to employ methods for concept and relation induction to predict plausible rules which are missing from a given ontology.

2 Related Work

Within the area of knowledge base completion, we can identify three classes of related work.

Methods for completing ABoxes and knowledge graphs. This class of methods focuses on finding missing facts. For example, a number of methods have been proposed that learn latent soft clusters of predicates to predict missing facts in relational data (Kok and Domingos 2007; Rocktäschel and Riedel 2016; Sourek *et al.* 2016). Within the area of knowledge graph completion, the most popular strategies are based on embedding relations and entities in a low-dimensional vector space, e.g. by modelling binary relations as vector translations (Bordes *et al.* 2013), or on identifying types of paths in the knowledge graph which are predictive of a given relationship (Gardner *et al.* 2014). Another possible strategy to find missing facts consists in extracting them from natural language statements (Mintz *et al.* 2009; Riedel *et al.* 2010; West *et al.* 2014). Most relevant for our work, some authors have also looked at predicting missing facts by modelling concepts in some underlying feature space. For example, Neelakantan and Chang (2015) represent each Freebase entity using a combination of features derived from Freebase itself and from Wikipedia, and then use a max-margin model to identify missing types. In (Bouraoui *et al.* 2017), description logic concepts were modelled as Gaussians in a vector space embedding.

Methods for learning rules from instances. In this paper, our focus is on predicting rules without using any database of facts (e.g. ABox assertions), which is motivated by the fact that for many useful ontologies no such database is available. However, when a sufficiently large database of facts is given, methods from Inductive Logic Programming (Bühmann *et al.* 2016), or based on Formal Concept Analysis (Baader *et al.* 2007) or Association Rule Mining (Völker and Niepert 2011), can be used to construct plausible rules. Such rules make explicit some of the regularities that are observed among the given facts, beyond those which are already encoded in the ontology.

Methods for completing rule bases. The problem of predicting plausible missing rules for a given rule base has not yet received much attention. In (Schockaert and Prade 2013), methods for completing propositional rule bases have been proposed, based on interpolation and analogical reasoning, but they were only studied from a theoretical point of view. Moreover, the methods proposed there require background knowledge, such as a betweenness relation in the case of interpolation, which is not readily available. In this paper, we avoid this issue by relying on word embeddings.

The idea of similarity based reasoning, as a general strategy for extending (the applicability of) rule bases, has been explored in a number of ways. For example, (Beltagy *et al.* 2013) uses Markov logic to consider defeasible rules of the form $\mathit{cucumber}(X) \rightarrow \mathit{zucchini}(X)$, to encode the intuition that many rules about cucumbers also apply to zucchinis. Conceptually this achieves a kind of similarity-based rule base completion (e.g. we may imagine adding a rule about zucchinis for each rule we have about cucumbers), although the extended knowledge base is not explicitly constructed.

Along similar lines, there have been a few proposals to extend logic programming with a soft unification mechanism, where a given rule is triggered, to some degree, if a formula is satisfied which is similar to the body of that rule, either based on a given similarity structure (Medina *et al.* 2004) or by similarity degrees which are induced from a vector space embedding (Rocktäschel and Riedel 2017).

3 Background

Ontologies express structured knowledge about the concepts, properties and relations of a given domain. Description logics and existential rules are the two main logical frameworks underlying ontology languages. While description logics are most often used in practice, in this paper we will consider existential rules, as this will simplify the presentation. Note however that the method we present in this paper could be straightforwardly applied to description logic axioms as well. Here we briefly recall the syntax of existential rules; for a comprehensive overview of this framework we refer to (Baget *et al.* 2011).

The syntax of existential rules is defined over a vocabulary of a finite set of relations (i.e. predicates) and an infinite set of constants. An *existential rule* is a first-order rule of the following form:

$$r_1(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y} \,.\, s_1(\mathbf{z_1}) \wedge ... \wedge s_m(\mathbf{z_m}) \quad (1)$$

Here $\mathbf{x_1}, ..., \mathbf{x_n}, \mathbf{y}, \mathbf{z_1}, ..., \mathbf{z_m}$ are tuples of variables and for each $i \in \{1, ..., m\}$ we have $\mathit{vars}(\mathbf{z_i}) \subseteq \mathit{vars}(\mathbf{x_1}) \cup ... \cup \mathit{vars}(\mathbf{x_n}) \cup \mathit{vars}(\mathbf{y})$, where we write $\mathit{vars}(\mathbf{x})$ for the set of variables appearing in the tuple $\mathbf{x}$. An example of existential rule is $\mathit{sibling}(x_1, x_2) \rightarrow \exists y.\ \mathit{parentOf}(y, x_1) \wedge \mathit{parentOf}(y, x_2)$ where *sibling* and *parentOf* are predicates and x_1, x_2 and y are variables. Note that the tuple $\mathbf{y}$ may be empty, which means that rules without an existential quantifier (e.g. $\mathit{sibling}(x_1, x_2) \rightarrow \mathit{sibling}(x_2, x_1)$) are also special cases of existential rules. In fact, the occurrence of existential quantifiers is quite rare in most real-world ontologies. In line with terminology from logic programming, the antecedent and consequent of an existential rule are sometimes referred to as the body and head respectively. An existential rule of the form $\top \rightarrow r(\mathbf{x})$ is called a fact, and a set of facts is sometimes called a database. An ontology is defined as a set of existential rules, which are usually assumed to be non-fact rules. The variables in an existential rule are implicitly assumed to be universally quantified. Standard notions such as consistency and entailment are then defined in the usual way.

4 A Model for Rule Induction

Throughout this section, we let $\mathcal{R}$ be a set of existential rules. The problem we consider is to identify rules ρ which are not entailed by $\mathcal{R}$, but are nonetheless plausible. As already mentioned in the introduction, our strategy is to consider rule templates, and to characterize the kind of relations that fit these templates based on vector space representations of these relations. In Section 4.1, we first explain what kind of templates are considered. Subsequently, in Section 4.2, we discuss in more detail how vector space representations can be used to represent relations. Then Sections 4.3 and 4.4 explain our induction models, respectively for unary and for binary templates. Finally, Section 4.5 discusses our overall approach to making predictions.

4.1 Rule Templates

We will consider two kinds of templates, which respectively replace one and two occurrences of relations by a second-order variable. Specifically, let ρ be an existential rule of the form (1). Then ρ induces the following unary templates:

$$\star(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y} \,.\, s_1(\mathbf{z_1}) \wedge ... \wedge s_m(\mathbf{z_m})$$
$$...$$
$$r_1(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y} \,.\, s_1(\mathbf{z_1}) \wedge ... \wedge \star(\mathbf{z_m})$$

We will write $\tau(\star)$, or simply τ, to denote a given unary template, where $\tau(r)$ then corresponds to the rule that is obtained when instantiating the second-order variable $\star$ with the relation r. Let us furthermore write $\mathcal{T}_1(\rho)$ for the set of all unary templates that can be obtained from the rule ρ, and let $\mathcal{T}_1(\mathcal{R}) = \bigcup_{\rho \in \mathcal{R}} \mathcal{T}_1(\rho)$. Similarly, the rule ρ induces the following binary templates:

$$\star\,(\mathbf{x_1}) \wedge \bullet\,(\mathbf{x_2}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge s_m(\mathbf{z_m})$$
$$...$$
$$\star\,(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge \bullet\,(\mathbf{z_m})$$
$$...$$
$$r_1(\mathbf{x_1}) \wedge ... \wedge r_n(\mathbf{x_n}) \rightarrow \exists \mathbf{y}.s_1(\mathbf{z_1}) \wedge ... \wedge \star\,(\mathbf{z_{m-1}}) \wedge \bullet\,(\mathbf{z_m})$$

where $\star$ and $\bullet$ are second-order variables. We write $\mathcal{T}_2(\rho)$ for the set of all binary templates that can be obtained from ρ and $\mathcal{T}_2(\mathcal{R}) = \bigcup_{\rho \in \mathcal{R}} \mathcal{T}_2(\rho)$. Similar as for unary templates, we write $\tau(\star, \bullet)$ to refer to a template, and $\tau(r, s)$ to the rule that is obtained by instantiating the second-order variables $\star$ and $\bullet$ by r and s respectively.

For a given unary template τ, we write $\pi(\mathcal{R}, \tau)$ for the set of relations that satisfy the templates in $\mathcal{R}$, i.e.:

$$r \in \pi(\mathcal{R}, \tau) \Leftrightarrow \tau(r) \in \mathcal{R}$$

Note that all relations in $\pi(\mathcal{R}, \tau)$ will have the same arity, which we will also refer to as the arity of the template τ. Similarly, for a binary template τ, $\pi(\mathcal{R}, \tau)$ represents the set of relation pairs that lead to a rule from $\mathcal{R}$, i.e.:

$$(r, s) \in \pi(\mathcal{R}, \tau) \Leftrightarrow \tau(r, s) \in \mathcal{R}$$

We now illustrate these notions in the following example.

Example 1. *Let us consider the following set of rules $\mathcal{R}$:*

$$\textit{livesIn}(X, Y) \wedge \textit{bat}(X) \rightarrow \textit{cave}(Y)$$
$$\textit{livesIn}(X, Y) \wedge \textit{duck}(X) \rightarrow \textit{pond}(Y)$$

Then $\mathcal{T}_1(\mathcal{R})$ contains the following unary templates:

$$\begin{aligned}
\tau_1(\star) &= \star(X, Y) \wedge \textit{bat}(X) \rightarrow \textit{cave}(Y) \\
\tau_2(\star) &= \star(X, Y) \wedge \textit{dolphin}(X) \rightarrow \textit{sea}(Y) \\
\tau_3(\star) &= \textit{livesIn}(X, Y) \wedge \star(X) \rightarrow \textit{cave}(Y) \\
\tau_4(\star) &= \textit{livesIn}(X, Y) \wedge \textit{bat}(X) \rightarrow \star(Y) \\
\tau_5(\star) &= \textit{livesIn}(X, Y) \wedge \star(X) \rightarrow \textit{pond}(Y) \\
\tau_6(\star) &= \textit{livesIn}(X, Y) \wedge \textit{duck}(X) \rightarrow \star(Y)
\end{aligned}$$

while $\mathcal{T}_2(\mathcal{R})$ contains the following binary templates:

$$\begin{aligned}
\tau_7(\star, \bullet) &= \star(X, Y) \wedge \bullet(X) \rightarrow \textit{cave}(Y) \\
\tau_8(\star, \bullet) &= \star(X, Y) \wedge \bullet(X) \rightarrow \textit{pond}(Y) \\
\tau_9(\star, \bullet) &= \star(X, Y) \wedge \textit{bat}(X) \rightarrow \bullet(Y) \\
\tau_{10}(\star, \bullet) &= \star(X, Y) \wedge \textit{duck}(X) \rightarrow \bullet(Y) \\
\tau_{11}(\star, \bullet) &= \textit{livesIn}(X, Y) \wedge \star(X) \rightarrow \bullet(Y)
\end{aligned}$$

Then we have e.g.:

$$\begin{aligned}
\pi(\mathcal{R}, \tau_1) &= \{\textit{livesIn}\} \\
\pi(\mathcal{R}, \tau_7) &= \{(\textit{livesIn}, \textit{bat})\} \\
\pi(\mathcal{R}, \tau_{11}) &= \{(\textit{bat}, \textit{cave}), (\textit{duck}, \textit{pond})\}
\end{aligned}$$

Our approach will be based on characterizing the commonalities of the relations in $\pi(\mathcal{R}, \tau)$. However, in some cases the templates we obtain might be too general for such characterizations to be meaningful. A typical example are subsumption rules such as $\textit{orange}(X) \rightarrow \textit{citrusFruit}(X)$, which gives rise to the binary template $\tau(\star, \bullet) = \star(X) \rightarrow \bullet(X)$. The instances of this template may have little or nothing in common, hence it would not be effective as a basis for induction. Therefore, we will also consider two kinds of restricted templates.

First, we consider *typed rule templates*, in which the possible instantiations of the second-order variables are limited to relations that are subsumed by a given relation. Let $\tau(\star, \bullet)$ be a binary template, and let r and s be two relations. Then $\tau(\star_{\downarrow r}, \bullet_{\downarrow s})$ is a typed template, whose instances are defined as follows: we have $(r_0, s_0) \in \pi(\mathcal{R}, \tau(\star_{\downarrow r}, \bullet_{\downarrow s}))$ if the following three conditions are satisfied:

1. $(r_0, s_0) \in \pi(\mathcal{R}, \tau(\star, \bullet))$
2. $\mathcal{R} \models r_0(X_1, ..., X_k) \rightarrow r(X_1, ..., X_k)$
3. $\mathcal{R} \models s_0(X_1, ..., X_l) \rightarrow s(X_1, ..., X_l)$

with k and l the arity of the relations r and s respectively.

The second kind of restricted templates we consider are based on constraining the name of the relations. This is motivated by the fact that relations whose name has the same suffix or prefix often have something in common (e.g. *rugbyPlayer*, *tennisPlayer*, *baseballPlayer*). If a naming convention is used which allows us to easily split relation names into meaningful constituents, we can easily restrict templates to relations that have a name with a particular suffix or prefix. In particular, we write $\tau(\star^{-str_1}, \bullet^{-str_2})$ for the

name-constrained restriction of $\tau(\star, \bullet)$ in which $\star$ can only be instantiated by relations whose name ends with str_1 and $\bullet$ can only be instantiated by relations whose name ends with str_2, and similar for prefixes.

4.2 Vector Space Representations

Given a template τ, and the knowledge that $\tau(r_1), ..., \tau(r_n)$ are valid rules, the main inference task we consider is to identify relations s for which $\tau(s)$ is a plausible rule, and similar for binary templates. To this end, we will use two types of vector space representations.

Word embeddings. First, we will use a pre-trained word embedding. Word embeddings allow us to exploit lexical background knowledge, intuitively enabling us to make predictions based on the idea that relations with similar names tend to have similar properties. To this end, we first tokenize the relation name using a small set of simple heuristics, based on standard ontology naming conventions. For instance, *ProfessionalRugbyPlayer* would be converted into a list of three words: *professional*, *rugby* and *player*. A relation r corresponding to the list of words $w_1, ..., w_n$ is then represented as the vector $v_r^w = \frac{1}{n}(\mathbf{w_1} + ... + \mathbf{w_n})$, where we write $\mathbf{w_i}$ for the vector representation of word w_i in the word embedding. While simply averaging the word vectors might seem naive, this strategy is known to be surprisingly effective for short texts (Hill *et al.* 2016).

Matrix factorization for unary templates. In addition to using the relation names, we can also derive information about the similarity of relations from the given rule base itself. Here the intuition is that relations which already appear in similar rules from the rule base should be considered to be similar. To implement this intuition, we will apply the idea from the AnalogySpace model (Speer *et al.* 2008). The aim of that model is to learn low-dimensional vector space representation of the concepts in the ConceptNet knowledge graph, by factorizing a matrix whose rows are concepts and whose columns are properties of these concepts. Adapted to our context, we start with a matrix M_1 whose rows are the unary templates from $\mathcal{T}_1(\mathcal{R})$, possibly extended by some of their typed or name-constrained variants. For efficiency reasons, we only include those templates τ for which $|\pi(\mathcal{R}, \tau)| \geq 2$. The columns of M_1 correspond to the relations from $\mathcal{R}$. There is a 1 in the row for τ and column for r iff $r \in \pi(\mathcal{R}, \tau)$, and a 0 otherwise.

The AnalogySpace method is based on the Singular Value Decomposition (SVD) of $M_1 = U\Sigma V^T$. In particular, the diagonal matrix Σ is replaced by the matrix Σ_k, in which all but the first k diagonal elements are replaced by 0. Using Σ_k, we can compute the approximation[1] $M_1' = U\Sigma_k V^T$ of the matrix M_1. Entries in M_1' which are close to 1, but which were 0 in M_1 then correspond to plausible rules which are missing from $\mathcal{R}$, i.e. if the entry on the row of template τ and the column of relation r is close to 1, then we may conclude that $\tau(r)$ is a plausible rule. This strategy was empirically found to work well in (Speer *et al.* 2008) for finding plausible missing links in ConceptNet, but has not yet been considered for finding plausible rules. We will evaluate this strategy as one of our baselines in Section 5.

[1] A well-known property of SVD is that M_1' is the rank-k matrix which is most similar to M_1, in terms of the Frobenius norm.

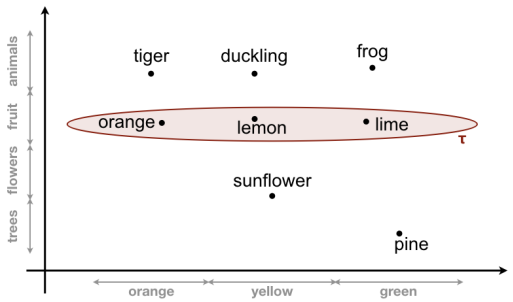

Figure 1: Modelling templates as ellipsoidal regions.

In our model, we will use the SVD decomposition of M_1 in another way. In particular, we can use Principal Component Analysis (PCA) to obtain a low-dimensional representation of the relations which maximally preserves the information encoded in M_1. To this end, we represent each relation r as the k-dimensional vector which is obtained by taking the first k columns of the row corresponding to r in the matrix $V\Sigma$. Let us denote this k-dimensional vector by $v_r^{\mathcal{R}}$. In our model, we will use the vectors v_r^w and $v_r^{\mathcal{R}}$ as two alternative representations of the relation r.

Matrix factorization for binary templates. We can also obtain a vector space representation of relation pairs, by applying PCA to the matrix M_2, whose rows are the binary templates (with at least two instances) and whose columns are relation pairs (which are instances of at least one template). Let us write $u_{r,s}^{\mathcal{R}}$ for the resulting k-dimensional vector representation of the relation pair (r, s). Similar as for unary templates, we will also use SVD to obtain a rank-k approximation of the matrix M_2 as a baseline strategy.

4.3 Unary Template Model

Intuition. Let us write v_r for the vector space representation of relation r (i.e. one of the two types of representations discussed in Section 4.2). Our main assumption is that the relations which satisfy some template τ are clustered together in the vector space. Whether this assumption is reasonable (for a given type of vector space) is an empirical question, which we will attempt to answer in our experimental evaluation below. However, a similar assumption was found to lead to good performance in (Bouraoui *et al.* 2017) for the task of ABox completion.

The most straightforward way to implement this assumption would be to learn a vector v_τ for the given template, and to assume that the probability that $\tau(r)$ is a valid rule can be expressed as a function of the similarity between v_τ and v_r. This closely corresponds to the strategy that was adopted in (Rocktäschel and Riedel 2017), although in a different setting. Note that the representation of a template in the vector space can then be viewed as a sequence of concentric spheres, containing the vectors of increasingly less similar relations. However, this relies on the rather unrealistic assumption that all dimensions of the vector space are equally

important. For example, it was found in (Mu *et al.* 2017) that some dimensions in popular word embedding models are far less informative than others. More generally, what typically matters is whether relations are similar with respect to particular facets. For example, *lemon* and *lime* are similar in most respects, but they have a different color. Accordingly, in some contexts, we may actually have to consider that *lime* is more similar to *frog* (because they are both green) than to *lemon*. To take this context-dependent nature of similarity into account, we will model templates using ellipsoidal regions in the vector space, instead of spheres. To illustrate this, Figure 1 shows a toy example with one dimension along which concepts are organized by color and one dimension along which concepts are organized by type. When modelling the template $\star(X) \rightarrow \textit{fruit}(X)$, only the latter really matters, leading to the ellipse shown in the figure.

To find a suitable (soft) ellipsoidal region for a given unary template τ, we will estimate a Gaussian distribution from the vector representations of the relations in $\pi(\mathcal{R}, \tau)$. This allows us to use the standard Bayesian machinery for estimating Gaussians, based on conjugate priors, which offers a convenient and principled way of avoiding overfitting. Conceptually, the resulting method for predicting plausible rules can be seen as the implementation of a form of commonsense reasoning which is known as interpolation (Schockaert and Prade 2013). In particular, the Gaussian modelling the template τ will offer us a convenient way of deciding whether a given relation r is sufficiently "between" the relations which are known to satisfy τ, to plausibly conclude that r satisfies τ as well.

Model description. Our aim is to evaluate the probability that a given template τ satisfies a relation r, knowing that it satisfies the relations $r_1, ..., r_n$. Using Bayes' rule we can express this as follows:

$$P(\tau(r) \,|\, v_r) = \lambda_\tau \cdot \frac{f(v_r \,|\, \tau(r))}{f(v_r)} \tag{2}$$

Here $f(. \,|\, \tau(r))$ is a Gaussian distribution modelling the relations satisfying the template τ. This distribution will be estimated from the vector representations of the relations $r_1, ..., r_n$. The distribution $f(.)$ expresses how likely the vector representation v_r itself is. It will be estimated as a Gaussian from the vector representations of the overall set of relations. In case the template τ is typed, however, $f(v_r)$ is estimated from the relations that have the correct type only, and similar for name-constrained templates. Finally, λ_τ is the prior probability that a given relation satisfies the template. It will act as a scaling factor.

Estimating Gaussians. Since $f(. \,|\, \tau(r))$ typically has to be modelled from a very small number of examples, we need to make some drastic regularity assumptions. In particular, we will make the common assumption that this Gaussian distribution has a diagonal covariance matrix (Vilnis and McCallum 2015). This means that we can evaluate this probability using a product of univariate Gaussians:

$$f(v_r \,|\, \tau(r)) = \prod_{i=1}^{m} G(x_i^r; \mu_i, \sigma_i^2)$$

where m is the number of dimensions in the vector space and we write x_i^r for the i^{th} coordinate of v_r. To estimate the parameters μ_i and σ_i^2 of these univariate Gaussians, we follow a Bayesian approach, i.e. rather than taking a single estimate, we take a weighted average based on a probability distribution over plausible values for these parameters. Compared to using maximum likelihood estimates, Bayesian estimation is more cautious and less prone to overfitting. A particular consequence is that templates with few instances are penalized, which will help our method to focus on the most reliable templates. Formally, the probability $G(x_i^r; \mu_i, \sigma_i^2)$ is then estimated as:

$$\int G(x_i^r; \mu, \sigma^2) NI\chi^2(\mu, \sigma^2 | \mu_0, \kappa_0, \nu_0, \sigma_0^2) d\mu d\sigma$$

where $NI\chi^2$ is the normal inverse χ^2 distribution, which is the standard conjugate prior of the Gaussian distribution. It encodes which values of the parameters μ and σ^2 are likely, given that the i^{th} coordinate of the vectors $v_1, ..., v_n$ is assumed to have been generated from that distribution, and possibly some prior information. In our setting, we will not assume that any prior information is given, in which case a flat prior can be used. It can be shown that the integral then evaluates to (Murphy 2007):

$$t_{n-1}\Big(\overline{x_i}, \frac{(n+1)\sum_{j=1}^{n}(x_i^{r_j} - \overline{x_i})^2}{n(n-1)}\Big)$$

where $\overline{x_i} = \frac{1}{n}\sum_{j=1}^{n} x_i^{r_j}$ and t_{n-1} is the Student t-distribution with $n-1$ degrees of freedom. We refer to (Murphy 2007) for more details on the Bayesian estimation of Gaussian distributions. The probability $f(v_r)$ is estimated in the same way, but based on the set of all relations (of the considered type), rather than only those in $\pi(\mathcal{R}, \tau)$.

Estimating the prior. The prior λ_τ is estimated by maximizing the log-likelihood of the rules in $\mathcal{R}$. Let the arity of the template τ be a, and let $\mathfrak{R}_a$ be the set of all relations of arity a. We then choose the value of λ_τ that maximizes:

$$\sum_{r \in \pi(\mathcal{R},\tau)} \log P(\tau(r)|v_r) + \sum_{r \in \mathfrak{R}_a \setminus \pi(\mathcal{R},\tau)} \log(1 - P(\tau(r)|v_r))$$

where $P(\tau \,|\, v_r)$, for a given choice of λ_τ, is evaluated as in (2). Note that this estimation of λ_τ relies on a closed world assumption, i.e. it is based on the assumption that τ only applies to the relations in $\pi(\mathcal{R}, \tau)$. Clearly this is not realistic. In fact, our rule completion method is motivated by the fact that some rules in $\mathcal{R}$ are missing. However, this simplifying assumption is needed because we do not have negative examples (i.e. relations for which it is given that the template does not apply). As a consequence, the value of λ_τ may be lower than it should be. However, this is typically not a problem, as it simply means that the predictions we make might be more cautious then they need to be. As a second simplification, in the case of large rule bases, the second summation will be restricted to a sample of $\mathfrak{R}_a \setminus \pi(\mathcal{R}, \tau)$ for computational reasons. A close approximation to this summation can be obtained by selecting the elements from $\mathfrak{R}_a \setminus \pi(\mathcal{R}, \tau)$ whose vector representation is closest to the mean of the Gaussian $f(. \,|\, \tau(r))$ (e.g. using a k-d tree).

4.4 Binary Template Model

Intuition. Like the unary templates, binary templates will also be modelled using Gaussian distributions. In the case of binary templates, however, there will be several Gaussians that are used in combination. In particular, we will learn (i) a Gaussian to model the kind of relations that may instantiate $\star$ (ii) a Gaussian to model the kind of relations that many instantiate $\bullet$, (iii) a Gaussian over the set of vector translations $v_s - v_r$ of valid instances (r, s) of the template, and, in case the SVD based vector representations are used, (iv) a Gaussian in the vector space of relation pairs. A model based on (i)–(iii) was already found to perform well for the task of relation induction in (Bouraoui *et al.* 2018), but it will here be adapted for the task of rule induction.

Model description. The probability $P(\tau(r,s) \mid v_r, v_s, u_{r,s})$ that a relation pair (r, s) satisfies the binary template τ is estimated as follows:

$$\lambda_\tau \cdot \frac{f(v_r|\tau(r,\bullet))}{f(v_r)} \cdot \frac{f(v_s|\tau(\star,s))}{f(v_s)} \cdot \frac{f(v_s - v_r|\tau(r,s))}{f(v_s - v_r|\tau(r,\bullet),\tau(\star,s))} \cdot f(u_{r,s}|\tau(r,s))$$

The scaling parameter λ_τ and the probabilities $f(v_r)$ and $f(v_s)$ are estimated similarly as in the unary template model. The probability $f(v_r|\tau(r,\bullet))$ represents how likely the vector representation v_r is, given that there exists some relation t such that $\tau(r,t)$ is a valid rule. It is estimated similarly to how we estimated $f(v_r \mid \tau(r))$ in the unary template model. The probability $f(v_s|\tau(\star,s))$ is also estimated in a similar way, but based on the second arguments of the elements in $\pi(\mathcal{R},\tau)$. The probability $f(v_s - v_r|\tau(r,s))$ is again estimated similarly, but now based on the vector differences $v_{s_1} - v_{r_1}$ of the elements (r_1, s_1) of $\pi(\mathcal{R},\tau)$. Finally, the probability $f(v_s - v_r|\tau(r,\bullet),\tau(\star,s))$ is estimated as follows. Let $A = \{r_1, ..., r_k\}$ and $B = \{s_1, ..., s_l\}$ respectively be the relations that occur as a first and as a second argument in the elements of $\pi(\mathcal{R},\tau)$. Each r_i is paired with a randomly selected element s'_i from B. Then we estimate $f(v_s - v_r|\tau(r,\bullet),\tau(\star,s))$ like $f(v_s - v_r|\tau(r,s))$, but by using the vector differences $s'_1 - r_1, ..., s'_k - r_k$ instead.

Note that the vectors $u_{r,s}$ are only available when using the vector representations obtained by PCA. In the variant of this model where we use vectors from word embeddings instead, the factor $f(u_{r,s}|\tau(r,s))$ is simply dropped.

4.5 Making Predictions

To estimate the probability that a rule ρ is valid, we first determine the corresponding set of unary templates $\mathcal{T}_1(\rho) = \{\tau_1, ..., \tau_k\}$ and binary templates $\mathcal{T}_2(\rho) = \{\tau_{k+1}, ..., \tau_l\}$. For each unary template τ_i let r_i be the relation for which $\rho = \tau_i(y_i)$, $1 \leq i \leq k$. Similarly, for a binary template τ_i we write $\rho = \tau_i(r_i, s_i)$, $k+1 \leq i \leq l$. The overall probability is then obtained by aggregating the probabilities obtained from both vector space representations, for each template, and then maximizing the resulting probabilities:

$$P(\rho \mid \mathcal{R}) = \max\Big(\max_{1\leq i\leq k} P(\tau_i(r_i)|v^w_{r_i}, v^{\mathcal{R}}_{r_i}), \max_{k+1\leq i\leq l} P(\tau_i(r_i,s_i)|v^w_{r_i}, v^w_{s_i}, v^{\mathcal{R}}_{r_i}, v^{\mathcal{R}}_{s_i}, u^{\mathcal{R}}_{r_i,s_i})\Big)$$

where $P(\tau_i(r_i)|v^w_{r_i}, v^{\mathcal{R}}_{r_i})$ is evaluated as:

$$\mu P(\tau_i(r_i)|v^{\mathcal{R}}_{r_i}) + (1-\mu)P(\tau_i(r_i)|v^w_{r_i})$$

with $\mu \in [0,1]$ a parameter controlling the relative importance of the two types of vector space representations. Similarly $P(\tau_i(r_i,s_i)|v^w_{r_i}, v^w_{s_i}, v^{\mathcal{R}}_{r_i}, v^{\mathcal{R}}_{s_i}, u^{\mathcal{R}}_{r_i,s_i})$ is evaluated as:

$$\mu P(\tau_i(r_i,s_i)|v^{\mathcal{R}}_{r_i}, v^{\mathcal{R}}_{s_i}, u^{\mathcal{R}}_{r_i,s_i}) + (1-\mu)P(\tau_i(r_i,s_i)|v^w_{r_i}, v^w_{s_i})$$

5 Experimental Results

In this section, we experimentally analyze the performance of our method. As knowledge bases we consider several well-known *OWL* ontologies, which we converted to existential rule bases. In particular, we consider two large-scale open-domain ontologies: SUMO and OpenCyc. We also test the performance on a number of smaller domain-specific ontologies: Wine ontology, Economy, Transport and Vicodi. Before converting each *OWL* ontology to a rule base $\mathcal{R}$, we use the pellet reasoner to compute the set of inferred axioms (subclasses, equivalent classes, sub-object properties and equivalent object properties), and we add the corresponding rules to $\mathcal{R}$ as well. As word embedding, We used a standard pre-trained 300-dimensional, which was learned using Skip-gram on the 100B words Google News corpus.

To evaluate the performance of different methods, we split the considered rule base into training and test sets. After splitting the rule base, we remove from the test set all rules that can be deduced from the training set. To evaluate our model, we will also need negative examples, in addition to the positive examples from the test set. Following a common practice in the context of knowledge base completion, we will generate a number of synthetic negative examples, which we will call distractor rules. In particular, following the strategy used in (Vylomova *et al.* 2016) for evaluating relation induction models, for each correct test rule *body* $\rightarrow$ *head*, we first add *head* $\rightarrow$ *body* as a distractor. Second, we also add one distractor rule of the form *body* $\rightarrow$ *head*$'$ and one distractor rule of the form *body*$'$ $\rightarrow$ *head*, where *body*$'$ and *head*$'$ are randomly selected from the heads and bodies that occur in the ontology. Before adding the distractor rules to the test set, we verify that they do not occur in the training or test set. While this does not guarantee that all distractors are invalid rules, we can expect this to be the case for the vast majority of them. To split the rule bases into training and test rules, we use 10-fold cross validation.

The considered task can be evaluated as a ranking task or as a classification task. When we consider it as a ranking task, the aim is to rank the correct test rules higher than the distractor rules. To evaluate the quality of the rankings produced by the different methods, we use precision at n (P@n), which is simply the percentage of the n highest ranked rules that correspond to correct test rules. We can also consider a classification task, i.e. for each rule in the test data decide whether it is a correct test rule or a distractor, where we report the F1 score.

To set the parameters of our model (and the baselines), we select 10% of the training data as validation data, and only

Table 1: Overview of experimental results.

		SUMO	Cyc	Wine	Vico	Trans	Eco
AS	F1	0.43	0.45	0.49	0.42	0.43	0.46
AS	P@10	0.51	0.54	0.53	0.46	0.49	0.51
AS	P@100	0.42	0.43	n/a	n/a	n/a	n/a
VS-$\mathcal{R}$	F1	0.40	0.43	0.47	0.46	0.48	0.50
VS-$\mathcal{R}$	P@10	0.47	0.46	0.50	0.49	0.51	0.54
VS-$\mathcal{R}$	P@100	0.41	0.39	n/a	n/a	n/a	n/a
VS-*w*	F1	0.42	0.45	0.49	0.49	0.50	0.48
VS-*w*	P@10	0.50	0.49	0.52	0.51	0.54	0.52
VS-*w*	P@100	0.43	0.42	n/a	n/a	n/a	n/a
VS	F1	0.47	0.50	0.55	0.52	0.53	0.51
VS	P@10	0.54	0.52	0.59	0.53	0.56	0.58
VS	P@100	0.46	0.47	n/a	n/a	n/a	n/a
RI-$\mathcal{R}$	F1	0.51	0.51	0.57	0.51	0.55	0.56
RI-$\mathcal{R}$	P@10	0.57	0.61	0.68	0.56	0.63	0.62
RI-$\mathcal{R}$	P@100	0.49	0.50	n/a	n/a	n/a	n/a
RI-*w*	F1	0.53	0.52	0.58	0.52	0.54	0.58
RI-*w*	P@10	0.60	0.62	0.67	0.59	0.61	0.62
RI-*w*	P@100	0.51	0.53	n/a	n/a	n/a	n/a
RI-UT	F1	0.51	0.52	0.53	0.54	0.58	0.57
RI-UT	P@10	0.54	0.54	0.60	0.58	0.61	0.62
RI-UT	P@100	0.49	0.51	n/a	n/a	n/a	n/a
RI-BT	F1	0.43	0.47	0.50	0.48	0.52	0.52
RI-BT	P@10	0.51	0.51	0.53	0.52	0.57	0.59
RI-BT	P@100	0.45	0.43	n/a	n/a	n/a	n/a
RI-WT	F1	0.50	0.48	0.51	0.50	0.52	0.53
RI-WT	P@10	0.53	0.50	0.58	0.53	0.57	0.59
RI-WT	P@100	0.47	0.49	n/a	n/a	n/a	n/a
RI	F1	0.56	0.54	0.61	0.58	0.62	0.63
RI	P@10	0.62	0.63	0.72	0.67	0.68	0.68
RI	P@100	0.54	0.55	n/a	n/a	n/a	n/a

use the remaining 90% for training the model. This validation data is used for selecting the parameter μ and for choosing the number of dimensions in the vector space representations $v_r^{\mathcal{R}}$ (chosen from $\{10, 25, 50, 100\}$). For the classification experiments, we also tune a threshold on the probability for a rule to be predicted as valid.

In the following we will refer to our model as **RI** (for Rule Induction). To better understand the impact of each component, we will also consider the following variants: **RI-$\mathcal{R}$** only uses the vector representations obtained using PCA and **RI-*w*** only uses vector representations from the word embedding, **RI**-UT only uses unary templates, **RI**-BT only uses binary templates, and **RI**-WT is our full model but without using restricted templates. We will also show results for two baselines. First, we will use the AnalogySpace model applied to unary rule templates, as described in Section 4.2 (**AS**). When used in a classification setting, we tune a threshold on the values of entries from M_1' above which the corresponding rule is considered valid. Second, we will use a similarity based model (**VS**). Given a template τ, we then learn a template vector v_τ which is the average of the vectors of the relations that satisfy this template, and then we tune a threshold on the similarity between this vector and the relation vectors to make predictions. To make the results comparable to those for **RI**, we represent each relation using the concatenation of its representation from the word embedding and from the PCA space. We also consider the variants **VS-$\mathcal{R}$** and **VS-*w***, which respectively only use the PCA space and the word embedding. This baseline will allow us to assess the benefit of using elliptical rather than spherical regions for characterizing templates.

An overview of the experimental results is presented in Table 1; note that no P@100 results are shown for the smaller ontologies, as the number of test rules is less than 100 in these cases. A number of conclusions can be drawn from the results. First, the proposed model clearly and consistently outperforms the baselines. Second, the PCA vector space and the word embedding space perform similarly, when used in isolation, but using the full model offers substantial further improvements. This illustrates the fact that both spaces effectively capture complementary information. Third, the **RI**-UT and **RI**-BT both perform clearly worse than the full model, showing that both types of templates are indeed necessary to achieve optimal results. Finally, the relatively poor performance of **RI**-WT is largely due the fact that most of the binary templates are very general, and therefore only become informative when we restrict them in a suitable way.

To illustrate how our model can outperform the similarity based strategy of **VS**, we give examples of rules that our model was able to predict, which go beyond similarity based reasoning. From the SUMO ontology, for instance, the unary template model correctly[2] predicts:

$$\mathit{Pipeline}(X) \rightarrow \mathit{Transitway}(X)$$

The template $\tau_1(\star) = \star(X) \rightarrow \mathit{Transitway}(X)$ was used to predict this rule, with $\pi(R, \tau_1) = \{\mathit{Airway}, \mathit{LandTransitway}, \mathit{Waterway}, \mathit{AirTransitway}\}$. Another example is the rule $\mathit{Sand}(X) \rightarrow \mathit{Soil}(X)$ which was predicted from $\tau_2(\star) = \star(X) \rightarrow \mathit{Soil}(X)$ and $\pi(R, \tau_2) = \{\mathit{Loam}, \mathit{Silt}, \mathit{Clay}\}$.

6 Conclusions

We have proposed a method for predicting plausible missing rules from a given ontology (i.e. a set of existential rules). The main underlying idea is to consider rule templates, which are second-order predicates whose instances correspond to rules. These templates allow us to approach the considered problem of rule induction as a particular kind of concept or relation induction problem. By considering both unary and binary rule templates, our method is able to implement several well-known commonsense reasoning strategies, including interpolation, similarity-based reasoning and analogical reasoning. From an application point of view, our method is easy to use, as the only required input is a rule base and a standard pre-trained word embedding.

Acknowledgments

Steven Schockaert was supported by ERC Starting Grant 637277.

[2]Correctly here means that this rule is included in the test data, but not in the training fragment of SUMO that was used for making the prediction.

References

Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In *Proc. IJCAI*, volume 7, pages 230–235, 2007.

Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of *SEM13*, pages 11–21, 2013.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proc. NIPS*, pages 2787–2795. 2013.

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert. Inductive reasoning about ontologies using conceptual spaces. In *Proc. AAAI*, pages 4364–4370, 2017.

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert. Relation induction in word embeddings revisited. In *Proc. COLING*, 2018.

Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. Dllearnera framework for inductive learning on the semantic web. *Journal of Web Semantics*, 39:15–24, 2016.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proc. EMNLP*, pages 397–406, 2014.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. Distributional vectors encode referential attributes. In *Proc. EMNLP*, pages 12–21, 2015.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proc. NAACL-HLT*, pages 1367–1377, 2016.

Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proc. ICML*, pages 433–440, 2007.

Ni Lao, Tom Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*, pages 529–539, 2011.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. Linguistic regularities in sparse and explicit word representations. In *Proc. CoNLL*, pages 171–180, 2014.

Jesús Medina, Manuel Ojeda-Aciego, and Peter Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy sets and systems*, 146:43–62, 2004.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proc. NAACL-HLT*, pages 746–751, 2013.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. ACL*, pages 1003–1011, 2009.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *CoRR*, abs/1702.01417, 2017.

Kevin Murphy. Conjugate Bayesian analysis of the Gaussian distribution. Technical report, University of British Columbia, 2007.

Arvind Neelakantan and Ming-Wei Chang. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proc. NAACL*, pages 515–525, 2015.

Daniel N Osherson, Edward E Smith, Ormond Wilkie, Alejandro Lopez, and Eldar Shafir. Category-based induction. *Psychological review*, 97(2):185–200, 1990.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

Jay Pujara, Danqi Chen, Bhavana Dalvi, and Sameer Singh Tim Rocktäschel, editors. *Proc. Workshop on Automated Knowledge Base Construction*, 2017.

Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proc. ECML/PKDD*, pages 148–163, 2010.

Tim Rocktäschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 45–50, 2016.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Proc. NIPS*, pages 3791–3803, 2017.

Steven Schockaert and Henri Prade. Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces. *Artif.Intell*, 202:86–131, 2013.

Gustav Sourek, Suresh Manandhar, Filip Zelezný, Steven Schockaert, and Ondrej Kuzelka. Learning predictive categories using lifted relational neural networks. In *Proc. ILP*, pages 108–119, 2016.

Robert Speer, Catherine Havasi, and Henry Lieberman. Analogyspace: reducing the dimensionality of common sense knowledge. In *Proc. AAAI*, pages 548–553, 2008.

Joshua B Tenenbaum and Thomas L Griffiths. Generalization, similarity, and bayesian inference. *Behavioral and Brain Sciences*, 24:629–640, 2001.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *Proceedings of the International Conference on Learning Representations*, 2015.

Johanna Völker and Mathias Niepert. Statistical schema induction. In *Proc. ESWC*, pages 124–138, 2011.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proc. ACL*, 2016.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proc. WWW*, pages 515–526, 2014.