Automated Service Selection using Knowledge Graphs

Muneera Bano[1], Alessio Ferrari[2], Didar Zowghi[1], Vincenzo Gervasi[3], and Stefania Gnesi[2]

[1] Faculty of Engineering and Information Technology, UTS, Sydney, Australia
muneera.bano@student.uts.edu.au, didar.zowghi@uts.edu.au
[2] CNR-ISTI, Pisa, Italy
{alessio.ferrari, stefania.gnesi}@isti.cnr.it
[3] Dipartimento di Informatica, Università di Pisa, Italy
vincenzo.gervasi@di.unipi.it

Abstract. With the huge number of services that are available online, requirements analysts face a *paradox of choice* (i.e., choice overload) when they have to select the most suitable service that satisfies a set of customer requirements. Both service descriptions and requirements are often expressed in natural language (NL), and natural language processing (NLP) tools that can match requirements and service descriptions, while filtering out irrelevant options, might alleviate the problem of choice overload faced by analysts. In this paper, we propose a NLP approach based on *Knowledge Graphs* that automates the process of service selection by *ranking* the service descriptions depending on their NL similarity with the requirements. To evaluate the approach, we have performed an experiment with 28 customer requirements and 91 service descriptions, previously ranked by a human assessor. We selected the top-15 services, which were ranked with the proposed approach, and found 53% similar results with respect to top-15 services of the manual ranking. The same task, performed with the traditional cosine similarity ranking, produces only 13% similar results. The outcomes of our experiment are promising, and new insights have also emerged for further improvement of the proposed technique.

Keywords: Service Selection, Requirements Engineering, Knowledge Graphs, Natural Language Processing.

1 Introduction

Software as a service (SaaS) offers reusability and flexibility for developing software while reducing time and costs for the developers [16]. However, this style of development has introduced new challenges, especially for analysts, when they have to find the best matching service for the requirements of all concerned stakeholders, while making a trade-off between cost, functional and quality requirements [5]. Identifying and selecting the adequate service is the most crucial step in service oriented software development [12]. It is considered a challenging task due to various reasons, as, e.g., mismatch in the level of abstraction and granularity of customer requirements and service descriptions, lack of context in services, and, in particular, *overchoice* (i.e., choice overload [18]), due to the large number of available services with similar characteristics [5]. Indeed, over the last decade, the number of third party services provided over the internet has increased substantially. This has exacerbated the situation for analysts when they have a huge set of services to evaluate, and select one that best serves the customer's requirements [3]. A significant amount of solutions have been proposed (e.g., [12, 21, 24]) that employ a variety of techniques for service identification, such as guidelines, patterns, business process models, and value analysis. However these approaches require application of formal methods to identify the adequate service for the requirements, and work only with a small and manageable number of services.

In this paper, we present a natural language processing (NLP) approach that uses Knowledge Graphs (KG) [9, 10] for dealing with the problem of overchoice [4]. The proposed approach has been evaluated in an experiment with a realistically large number of service descriptions – i.e., 91 descriptions retrieved from the Web. The approach was compared with a manual evaluation, and with a more traditional information retrieval approach based on cosine similarity [19]. The KG-based approach substantially outperforms the one based on cosine similarity in selecting the most relevant services, after filtering out irrelevant options.

The major contributions of this research are twofold: (a) a novel NLP method for substantially decreasing the number of available services found by search engines that match a set of requirements, and (b) supporting requirements analysts in reducing the time and effort needed for optimal service selection.

This paper is organized as follows. Sect. 2 provide pointers to related works. Sect. 3 provides an overview of the service selection task to be addressed, as well as the automated techniques employed in this paper. Sect. 4 presents the experimental evaluation performed and Sect. 5 discusses the results achieved, while Sect. 6 provides comments concerning the threats to validity that might have affected our work. Finally, Sect. 7 concludes the paper.

2 Related Works

Over the last decade, a significant amount of research has been conducted in providing tools, techniques and methods to deal with the task of service selection [14, 16], and various researchers have proposed to *automate* the service identification process. Among these works, the SeCSE project [23] has the aim to create free and open source methods, tools and techniques for system integrators and service providers. Their main objective is to provide a mechanism where discovered services can be used for improving and refining the initial requirements and thus bringing better alignment of requirements and business needs to the services and vice versa. IBM has proposed the SOMA [2] approach, which is an end-to-end software development method for service oriented solutions. The SENSORIA [22] project had the aim to develop a new approach for service oriented software engineering with existing theories, techniques and methods for ensuring correctness of the procedure and allowing a semi-automatic design process. The usage of *business process* (BP) models for service identification is another relevant line of research. Jamshidi *et al.* [13] focus on the service-oriented development with BP models, while Adam *et al.* [1] propose to use BPs for identifying the services at the right level of abstraction and granularity. These approaches make the assumption that the BP models will have complete details of their corresponding processes, which is hard to achieve. These existing approaches require a certain degree of formalism (e.g., XML, UML, BP models) for comparing the requirements and service specifications and do not operate with specifications in NL and with overload of choice. With a considerably huge number of available services, conversion of service specifications to formal notations is a time consuming task. Our proposed KG approach reduces the sample of available services to a manageable size without effort of formalisation, and within a short time span.

3 Computing Similarity

The "paradox of choice" in the case of service selection is due to the presence of too many service options among which the requirements analyst can choose to satisfy his/her requirements. To address this problem, our goal is to automatically discard those service options that are less relevant for the requirements, and present a reduced list of services, which can be browsed by the analyst to perform a selection within a more manageable set of choices. To this end, we wish to provide an automated technique that *ranks* the services according to their degree of satisfaction with respect to the set of requirements. Once a ranked list of services has been automatically produced, the analyst can discard the services that have lower ranking. The idea of the approach is that automated ranking can be performed by computing the *similarity* among the NL content of the requirements and the NL descriptions of the services. The underlying assumption is that if a requirement and a description of a service are semantically similar in terms of NL content, the service is likely to satisfy the requirement.

More formally, given a set of requirements $R = \{r_0, \ldots, r_m\}$, and a set of descriptions of services $S = \{s_0, \ldots, s_l\}$, we wish to rank the services according to their degree of matching with respect to the requirements. To this end, we compute the similarity σ of a requirement r_i with respect to a service description s_j. The overall ranking K of a service description s_j with respect to a set of requirements R is given by:

$$K(s_j, R) = \sum_{i=0}^{m} \sigma(r_i, s_j).$$

Given $|S|$ service descriptions, we can compute the ranking K of each service, and order the services according to K. Then, the analyst can discard the k services that have lower ranking, and manually analyse the $|S| - k$ services with higher ranking. In this paper, $\sigma(r_i, s_j)$ will be implemented by two functions, namely the Knowledge Graph similarity function, and the classic cosine similarity [19] function, which are described in the following sections.

3.1 Knowledge Graph Similarity

A Knowledge Graph [10] is a representation of a document set D in the form of a directed weighted graph $KG = \{V, E, w\}$, where V are nodes, E are edges and w is a weighting function. In our context, the set of documents D is composed of those documents that give the definitions of the concepts included in the requirements. For example, if – as in our experimental evaluation (Sect. 4) – the requirements concern an SMS gateway service, the documents in D will include a NL definition of what a SMS gateway is, a description of the standard protocols considered in the requirements (e.g., HTTP, SOAP), as well as other documents describing relevant concepts mentioned in the requirements. For a precise definition of E, V and w, the reader should refer to [10]. Here, we give the essential information required to understand the rest of the paper.

Nodes. Each node $v \in V$ in the KG is labeled with the morphological root of a relevant term contained in D. The relevant terms are those terms that are not part of the so-called *stopwords*, i.e., words which are common in English such as pronouns, prepositions, articles and conjunctions. The use of morphological roots (also called *stems*) implies that terms such as "communication" and "communicator" are both represented by the same node "communic". We call $\overline{T}$ the stems of the unique terms in D that are not stopwords.

Edges. Each directed edge in the graph is associated with the *co-occurrence* of two terms in the document set. We say that two terms co-occur when they appear *aside* each other in a sentence. Co-occurrence normally indicates semantic proximity among concepts [20], and therefore we use it to represent concepts relationships. The nodes of our graph are not labeled with terms, but with stems. Hence, the concept of co-occurrence is in this case related to stems. The edges $e = (v_i, v_j) \in E$ in our graph are all those couples of nodes such that the corresponding labels of the nodes co-occur in D. The direction of each edge represents the order in which the stems connected by the edge can appear in a sentence of the document set.

Weighting function. Each edge is labeled with a real positive number according to the weighting function w. The weighting function $w : E \rightarrow [0, 1]$ is the inverse function of the semantic relatedness of two stems. The more two stems connected by an edge co-occur in D, the lower the value returned by w.

An example excerpt of a KG extracted from the Wikipedia document for the term "SMS" is presented in Fig. 1. We see that the weight of the edges is lower for those stems that frequently occur together in the document (e.g., for $e =$ ('text', 'messag'), $w(e) = 0.058$), and have a tighter semantic connection in the domain. Higher weights (e.g., $e =$ ('protocol', 'SMS'), $w(e) = 0.5$) or no edge (e.g, between 'mobil' and 'short') occur when the semantic connection in the domain is weaker.

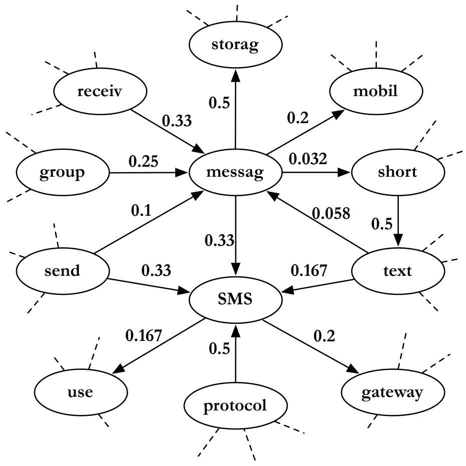

Fig. 1. Excerpt of a Knowledge Graph

KG-expansion. Given a term, we wish to expand it with the set of concepts associated to such term in the KG. To this end, we define the term-expansion function $I : \overline{T} \times KG(D) \to 2^{\overline{T}}$ as the function that associates the stem of a term to all the stems that are directly connected to such stem in the KG. More formally, given a stem $\bar{t}$ in $\overline{T}$, we define the term-expansion function as:

$$I(\bar{t}, KG(D)) = \{\bar{s} \in \overline{T} : \exists e \in E, e \in \{(\bar{t}, \bar{s}), (\bar{s}, \bar{t})\}, w(e) \leq \epsilon\} \cup \bar{t}$$

The value $\epsilon \in [0, 1]$ is used to discard the stems connected with low semantic proximity. If we apply such function to the term "SMS", we consider the KG in Fig. 1, and we set $\epsilon = 0.4$ such function will return the set ('SMS', 'messag', 'text', 'send', 'use', 'gateway'). The term-expansion function can easily be extended to the sets of non-stopwords stems $\overline{q} = \{\overline{q}_1, \ldots, \overline{q}_{|\overline{q}|}\}$ of a generic input document q. We call this function KG-expansion function $\mathcal{I}$, defined as:

$$\mathcal{I}(\overline{q}, KG(D)) = \bigcup_{j=1}^{|\overline{q}|} I(\overline{q}_j, KG(D))$$

Given a requirement or the description of a service – which can be both regarded as documents q – these can be represented through the sets of their stems. Stopword removal can be applied to represent them as $\overline{q}$. Afterwards, such sets can be expanded through the KG-expansion function, which basically extends the concepts included in the service description or requirement with the associated concepts in the KG.

KG-similarity. The KG-similarity function aims to compute the similarity between a requirement r_i and a service description s_j with the support of the KG. Such similarity function is a Jaccard similarity metric [19] based on the expansion of both the requirement and the service description through the function $\mathcal{I}$. We define the KG-similarity function between r_i and s_j as:

$$\sigma_{KG}(r_i, s_j) = \frac{|\mathcal{I}(\overline{r}_i, KG(D)) \cap \mathcal{I}(\overline{s}_j, KG(D))|}{|\mathcal{I}(\overline{r}_i, KG(D)) \cup \mathcal{I}(\overline{s}_j, KG(D))|}$$

In a sense, this similarity metric measures the proportion of stems that the extended version of a requirement and the extended version of a service description have in common, and it assumes that the greater is the number of shared stems, the higher is the chance of the requirement to be satisfied by the service description.

3.2 Cosine Similarity

Vector Space Model. In information retrieval [15], it is typical to represent documents according to the vector-space model [17], which has been also employed in requirements engineering to support the computation of the similarity among requirements [6, 8]. With this model, a natural language document q is regarded as a sparse vector $\boldsymbol{q} = \{q_{u_0}, \ldots, q_{u_h}\}$, where each vector component q_u is associated to a term u in the vocabulary of the vector space. Such vocabulary is made of all the terms included in the documents to be evaluated. In our case, the vocabulary is made of all the terms in the service descriptions.

The value of a component q_u is 0 if the term u does not appear in q, and is included in $(0, 1]$ if the term appears in q. The value of q_u scores the relevance of the term for the document q. Such relevance is normally computed by means of the $tf - idf$ score [15].

COS-similarity. The cosine similarity function aims to compute the similarity between a requirement r_i and a service description s_j within the vector space, whose vocabulary is made of all the terms included in the service descriptions. Each r_i and each s_j are first represented according to the vector space model as $\boldsymbol{r}_i$ and $\boldsymbol{s}_j$. Then, their similarity is evaluated according to the cosine similarity function σ_{COS} defined as:

$$\sigma_{COS}(r_i, s_j) = \frac{\boldsymbol{r}_i \cdot \boldsymbol{s}_j}{|\boldsymbol{r}_i||\boldsymbol{s}_j|}$$

Such similarity measures the cosine of the angle between the two vectors, and it assumes that the more relevant terms the requirement and the service description have in common, the higher is the chance of the requirement to be satisfied by the service description. In our implementation we have discarded stopwords and we have employed *stems* instead of terms in our vector space model. Therefore, in the definitions given above, each term u shall be considered a non-stopword stem $\overline{u}$.

4 Experimental Evaluation

The data used for the experiment was reused from the previously conducted case study on service selection [4]. The case was about an SMS gateway service selection, which enables Websites to send and receive text or multimedia

Table 1. Excerpt of the requirements set adopted in our experiments (top), and excerpts of two service descriptions (bottom).

ID	Requirement
0	Service supports outgoing text messages in Australia
1	Service should not have any hardware or SIM requirements
2	Service should be highly reliable with 99.9% message delivery
...	...
22	SMS transmission delay should be between 3 to 7 seconds
23	Service should offer contacts management
24	Service supports retrieval of send messages record
25	Service shows message delivery message
26	Service shows message delivery failure notification
27	Service should provide schedule message delivery

Messente
Send SMS online, SMS gateway API
Easiest group SMS messaging in the universe
Start sending SMS to your customers anywhere
in the world within 60 seconds
Send SMS to 200+ countries
No fees
...

Intel Tech
Why Choose Us?
Cloud Services
Lightening Fast Delivery Times
Amazing 24/7 Support
Reliable 100%
Uptime SLA
...

messages with simple invocation of the remote service API while hiding all the underlying technical and infrastructure details. Online searches resulted in 91 eligible options providing the SMS gateway services, which were to be evaluated against the 28 requirements in [4] (an excerpt is shown in Table 1 - top). The list of 91 services along with the links to their descriptions are available online at `http://goo.gl/CcguZM`. Two representative excerpts of NL service descriptions (according to their Websites at the time of the experiments) are reported at the bottom of Table 1.

The goal of this experiment was to evaluate to which extent the KG-similarity function and the COS-similarity function could be employed to discard the less relevant services, and therefore address the problem of overload of choice by reducing the list of service options that a requirement engineer has to evaluate.

4.1 Design and Execution

The experiment was carried out separately by the first and second author, referred in the following as subject 1 and subject 2, respectively. Fig. 2 shows the different approaches that have been applied in the experiment. The input of the experiment – equivalent for both subjects – was composed of the set of NL requirements $R = \{r_0, \ldots, r_{27}\}$, and the links to the main Web-page of the 91 services. The NL content of these Web-pages represented our service descriptions $S = \{s_0, \ldots, s_{90}\}$.

Subject 1 performed the service ranking task manually (approach M), and subject 2 performed the ranking task by first applying the KG-similarity function (approach KG), and then the COS-similarity function (approach COS). Approach KG includes a manual set-up approach, where the domain documents D have been retrieved and selected. Approach COS does not include a manual set-up and it is fully automatic. For simplicity, we refer to both approaches as

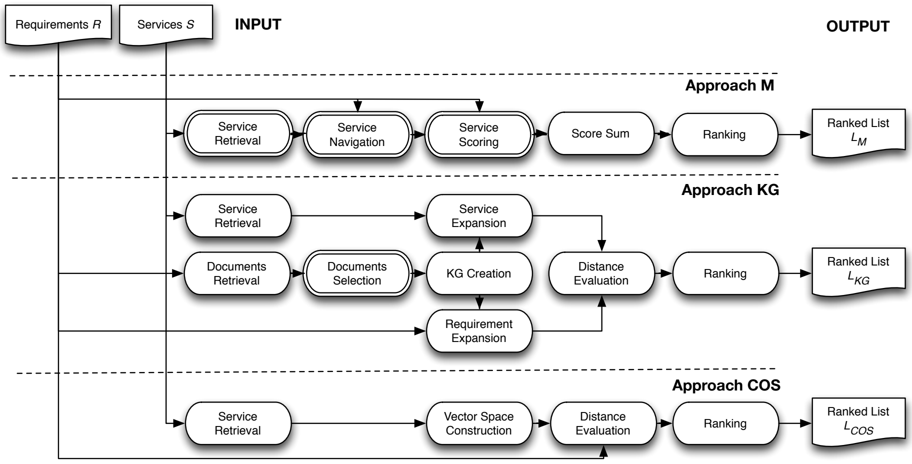

Fig. 2. Overview of the different treatments applied. Double-lined ellipses indicate manual activities.

"automated approaches". The result of each approach was an ordered list of services, ranked according to their degree of satisfaction with respect to the requirements. The ranking provided by means of approach M has been regarded as a *ground-truth* against which the automated approaches (KG and COS) have been tested in terms of ability in discarding the less relevant services.

Approach M. In approach M, subject 1 browsed through all the online service descriptions (**Service Retrieval** in Fig. 2) and gave a score to the services against each requirement (**Service Scoring**). Whenever the content of the main Web-page of the service description was not giving enough information to evaluate the degree of satisfaction with respect to the requirements, subject 1 navigated the Web-links provided in the main Web-page to gather more information about the service (**Service Navigation**). In a sense, the strategy followed by subject 1 was to use the content of the Web-pages (i.e., $S = \{s_0, \ldots, s_{90}\}$), and extend such content with additional information that could be found within the Web-links. For each requirement, a service was scored with three values i.e. 0= requirement not satisfied; 0.5= requirement partially satisified; 1= requirement completely satisfied. Then, the scores obtained by the service on each requirement were summed up to obtain a ranking score K for the service (**Score Sum**).The output of approach M was a ranked list of services L_M (**Ranking**), with associated a ranking score K computed according to the scoring schema described above. The overall time required to perform the evaluation with approach M was two weeks.

Approach KG. In approach KG, subject 2 automatically retrieved the NL content of the Web-pages of the services to compare such content (i.e., $S = \{s_0, \ldots, s_{90}\}$) with the requirements (**Service Retrieval**). Then, subject 2 selected a set of documents D to build the knowledge graph. Such set was selected by automatically downloading the textual content of the Wikipedia pages that were associated to the terms included in the requirements (**Documents Retrieval**). More specifically, for each term in the requirements, the Wikipedia page associated to that term was downloaded, when available. The downloaded documents were briefly reviewed to discard irrelevant pages (e.g., the Wikipedia page for "soap" was referring to the cleaning soap and not to the SOAP protocol). Additional domain documents were included in D for those technical terms that were considered more relevant, namely "HTTP", "PHP", "SMS gateway service", "SMS", "SOAP". In total, D included 28 documents, 23 automatically selected and 5 manually added. These two activities are referred in Fig. 2 with the task named **Documents Selection**. Finally, subject 2 performed the experiments with a Python implementation of the approach described in Sect. 3.1, which consists in building a knowledge graph from the documents (**KG Creation**), and then expanding requirements and services by means of the knowledge graph (**Requirement Expansion** and **Service Expansion**, respectively). The KG extracted from the documents resulted in $|V| = 6683$ nodes (i.e., individual stems) and $|E| = 37253$ edges (i.e., co-occurrences). Evaluation of the KG-similarity (**Distance Evaluation**) has been performed with $\epsilon = 0.5$, hence all the edges with weight 1 – i.e., associated to single co-occurrences – have been discarded. The output of the experiment was a ranked list of services L_{KG} (**Ranking**), with associated ranking score K computed by summing-up the contributions of the KG-similarity function on each requirement. The overall time required by approach KG was 25 minutes.

Approach COS. In approach COS, subject 2 used the service descriptions already retrieved with approach KG (**Service Retrieval**). Then, subject 2 performed the ranking task with a Python implementation of the approach described in Sect. 3.2, which consists in building the vector space (**Vector Space Construction**) and in evaluating the distance between services and requirements through the cosine similarity metric (**Distance Evaluation**). The output of the experiment was a ranked list of services L_{COS} (**Ranking**), with associated ranking score K computed by summing-up the contributions of the COS-similarity function. The overall time required by approach COS was 2 minutes.

4.2 Results

To evaluate the results of the experiments, we measured the "degree of correspondence" between the ranking of the *ground-truth* (approach M) and the rankings of approach KG and COS, in terms of services that have been considered less relevant – and therefore discarded – by the three approaches. The index adopted to evaluate such "degree of correspondence" is the accuracy, which, in our context, will be referred as *filtering accuracy*. Given the ranked list L_M of approach M, and given the ranked list L_A of one of the automated approaches, let O_M be the set of services discarded by approach M, and let O_A the set of services discarded by one of the automated approaches. Moreover, let k be the number of services discarded in both approaches (i.e., $k = |O_A| = |O_M|$). The filtering accuracy is:

$$\alpha = \frac{|O_M \cap O_A|}{k}.$$

The value of α has been computed by varying k in $[1, |S|]$, with $|S| = 91$, for our experiments. The idea of such evaluation is: if the requirements engineer discards the k less relevant services form L_A, how many of the services discarded belong to the k less relevant services of L_M? Since L_M comes from a human evaluation, high values of α gives confidence on the fact that the automated approach discards the same services that would be discarded by a human.

Moreover, we have also computed an index of accuracy that considers how many relevant services are retrieved by the automated approaches. We call this index *selection accuracy*, and we define the index as follows (Also in this case, we evaluate ρ by varying k in $[1, |S|]$):

$$\rho = \frac{|(L_M \setminus O_M) \cap (L_A \setminus O_A)|}{|S| - k}.$$

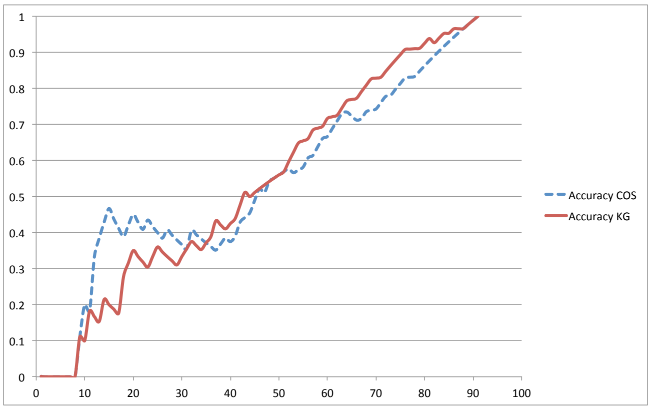

Fig. 3. Results for the filtering accuracy.

Fig 3 plots the filtering accuracy for approach KG and approach COS (α_{KG} and α_{COS}, respectively), for increasing values of k. Let us now suppose that the analyst wishes to filter out the 90% of the irrelevant services (top-right part of Fig. 3). We see that, to have $\alpha_{KG} > 0.9$ (i.e., a 90% filtering accuracy), we must have $k > 75$. This implies that the analyst can focus on the top-16 services, being confident that 90% of the irrelevant services have been correctly filtered. Slightly lower performances are achieved for α_{COS}. Indeed, the filtering accuracy is above 0.9 only for $k > 82$, which implies that the analyst has to discard 82 services from the ranked list, to be sure to filter out the 90% of irrelevant services. However, as visible by looking at the top-right part of Fig. 3, the ability of filtering the

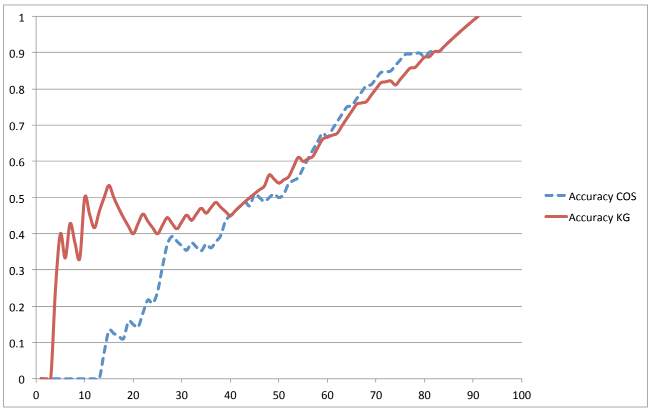

Fig. 4. Results for the selection accuracy.

irrelevant services for high values of k can be considered comparable for the two methods. It is also worth noting that, for low values of k, the filtering accuracy of approach COS is considerably higher than that of approach KG (see left part of Fig. 3). We argue that this might be due to an *inclusive* tendency of the KG-expansion function. Indeed, such function expands the concepts of a service description with additional concepts, and even though the description is rather poor in terms of requirement coverage, it could become richer thanks to the additional concepts included in its KG-based expansion. However, such inclusive nature appears to have a negative effect only for lower values of k. For high values of k (i.e., if we consider only the top-ranked services), the effect of such inclusive nature appears to be negligible with respect to the positive effect in providing contextual concepts, which seems to provide a more accurate requirement-to-service matching. Let us now consider the selection accuracy. Fig 4 plots the selection accuracy for approach KG and approach COS (ρ_{KG} and ρ_{COS}, respectively), for increasing values of $|S| - k$. From the plot, we see that, given a set of 15 top-ranked services (local maximum on the left part of the graph), approach KG is able to identify 53% of the relevant services, while approach COS identifies only the 13% of the relevant services. To assess the effectiveness of the approaches, it is useful to compare the selection accuracy on the top-15 results with a random predictor model. Such a model assumes to randomly select $k = 15$ items among the $|S| = 91$ items: the accuracy for such a model is $k/|S| = 0.16$. Hence, the selection accuracy is 16% for the random predictor, 53% for approach KG and 13% for approach COS. Therefore, approach KG outperforms the random predictor by 37%, while for approach COS the accuracy is even lower by 3% with respect to the random predictor. Note that having a local maximum of accuracy for $k = 15$, is an encouraging result in practical terms. Indeed, given a ranked list of results, as occurs for search engines, people tend to focus on the first page of the results (see, e.g., `https://chitika.com/google-positioning-value`), which normally displays

10 to 15 items. We conjecture that, for an analyst, 15 items can be considered a manageable set also in the case of service selection. To have a clear view of this result, it is useful to look at the tables of the top-15 services of the three approaches.

Table 2 (left) reports the top 15 services together with their rank according to the manual assessment, while Table 2 (center) reports the top 15 services obtained with approach KG. The 8 services in common in the two lists are highlighted in **bold**. The tangible results presented in Table 2, can give us confidence on the effectiveness of the method: if we consider a requirements analyst who searches for the best service that satisfy a set of requirements, he/she can have confidence that the top results of the approach will be likely to be suitable for its needs. Then, he/she can manually review the top results to select the best service.

Instead, the top-15 results obtained with approach COS (Table 2 (right)) show that only two of these results appear in Table 2 (left). Therefore, in a practical scenario, the analyst would have gone through the top-15 services, and found that really few of them were in line with his/her requirements.

Table 2. Ranking for approach M (left), KG (center), and COS (right).

Service	***K***
Intel Tech	27.5
Skebby	27.5
Direct SMS	27
Via SMS	27
Messente	26.5
Ready to SMS	26
Click Send	25
Bulk SMS	24.5
Clock Work SMS	24.5
Red Oxygen	24.5
Clickatell	24
SMS Broadcast	24
SMS Global	23.5
Essendex	23.5
Nexmo	23.5

Service	***K***
Messente	7.58
One API 4 SMS	6.25
Red Oxygen	5.84
Intel Tech	5.82
Click Send	5.75
Wave Cell	5.74
Cdyne	5.62
Budget SMS	5.61
Plivo	5.47
Clock Work SMS	5.46
SMS Global	5.35
M4U	5.25
TXT Nation	5.25
Nexmo	5.16
Bulk SMS	5.09

Service	***K***
Text Impact	2.45
Text Anywhere	2.42
SMS Roaming	2.35
Vodafone Multitext SMS	2.33
Developer Garden	2.31
Oventus	2.20
Free SMS Craze	2.19
SMS Country	2.15
Text Marks	2.10
Carousel SMS	2.07
Essendex	1.98
Text Local	1.91
Ausie SMS	1.89
Ready to SMS	1.82
GSMA	1.68

5 Discussion and Improvements

The major contribution of the study is the automation of the process of matching requirements against a big set of service descriptions by means of the KG approach. The KG approach has yielded 53% selection accuracy, and 90% filtering accuracy within a manageable reduced list of services. We argue that the accuracy of the results may have been affected by the procedure for service browsing. The KG approach only used the first page of the Websites for service description to match them against the requirements. Whereas in manual evaluation, the researcher went through further to seek more information to score the services (i.e., a **Service Navigation** task was introduced). A similar technique can be implemented in the KG approach to traverse the relevant links on the first page to retrieve further information.

Another issue to be addressed in the KG approach is associated to its coarse-grained similarity computation. Indeed, requirements and services might include relevant fine-grained constraints. Consider the excerpt of the "Messente" service description in Table 1, with respect to requirement 22: *"SMS transmission delay should be between* 3 *to* 7 *seconds"*. From the description, which tells *"Start sending SMS to your customers anywhere in the world within 60 seconds"*, we see that the service is able to send messages within 60 seconds. Therefore, it is likely to fail in satisfying requirement 22. Nevertheless, the "Messente" service is evaluated as the top-service for the KG approach. This result is due to two factors: (1) summing-up the values of the contributions of the different requirements to provide an overall ranking might hide those requirements for which the service is not the best; (2) our measure of similarity among a requirement and a description focuses on the "topic" of a requirement (e.g., the transmission delay), and does not consider the lower-level semantic dimension (e.g., the actual temporal constraint). The first issue can be easily addressed, by considering the similarity of the *single* requirement (i.e., the KG-similarity) with respect to the whole service, and avoiding to sum-up the contributions of each requirement. Therefore, the approach already include a way to spot out *outliers*, i.e., requirements not fully satisfied. Instead, to address the second issue, and achieve a finer grain similarity measure, specific heuristics have to be defined and integrated in the methodology.

A final aspect to discuss is the time required by the KG approach. Though 25 minutes are acceptable, it is still a rather high amount of time for an automated approach. The bottleneck of the approach is the **Service Expansion** task, which, in our experiments, required 17 minutes. Besides code-level optimization, given the higher efficiency of the COS approach, which required only 2 minutes in total, we argue that the two approaches could be combined as follows. The COS approach could be used for lower values of k, where its filtering accuracy is even higher than the KG approach. Then, the KG approach can be employed only on the pages that received higher ranking according to the COS approach. The computation of the exact threshold for the usage of one approach and the other (i.e., from which threshold value of k is preferable to use the KG approach) requires further studies with multiple data-sets.

6 Threats to Validity

In our experiments, we have used two treatments, namely approach KG and approach COS, to compare to a manual service selection treatment (approach M), considered as a reference baseline. As shown in Fig. 2, the input of each treatment was the same (i.e., requirements and services), and the treatments were evaluated according to a comparable output (i.e., a ranked list). Two performance measures have been employed to evaluate the results, namely the filtering accuracy and the selection accuracy. Moreover, we have also shown a humanunderstandable view of the results in the form of ranking tables. Therefore the design of the experiment provides confidence on the results achieved by approach KG. Concerning a possible researcher's bias, it is worth highlighting that the two researchers (subject 1 and 2) operated in parallel in two different institutions, and produced their evaluations independently.

The confounding variables that can affect the internal validity of the results could be attributed to the difference in the scoring procedure adopted by the manual and automated approaches. Furthermore, the retrieval technique used in the manual experiment was more comprehensive than that of the automated ones because, as we mentioned, more Web-pages were traversed to retrieve relevant information and has also impacted the results of the experiment. Though these two confounding variables could in principle threat the internal validity of our results, it is worth noting that, in the evaluation, the approaches were compared according to their input and output, which, by experimental design, are comparable. Hence, the effect of such variables is mitigated by the design of the experiment. The experiment design, data collection and execution are described in sufficient details to make it repeatable. However, the service descriptions that were retrieved from the online URL may change over time (e.g. in case of update or new release of the Web-site). Therefore, in that scenario, the replication with similar techniques may provide different results.

7 Conclusion

With increase in the number of services offering similar functionality, analysts face the problem of *overchoice* when they have to select one service against requirements. In this paper we have presented an approach that automates the process of matching requirements against service descriptions using *Knowledge Graphs* [10, 9]. Though the approach is promising to address the challenges of service selection, there is still room for improvement in the efficiency of the approach and accuracy of the results (see Sect. 5). Our future research will focus also on comparing the performance of the KG-similarity metric with the other existing approaches for computing NL requirements similarity (see, e.g., [7, 11]), to assess their effectiveness in the field of service selection.

References

1. Sebastian Adam, Norman Riegel, and Joerg Doerr. Deriving software services from business processes of representative customer organizations. In *SOCCER'08*, pages 38–45. IEEE, 2008.
2. Ali Arsanjani, Shuvanker Ghosh, Abdul Allam, Tina Abdollah, Sella Ganapathy, and Kerrie Holley. Soma: A method for developing service-oriented solutions. *IBM systems Journal*, 47(3):377–396, 2008.
3. Muneera Bano. Aligning services and requirements with user feedback. In *RE'2014*, pages 473–478, 2014.
4. Muneera Bano and Didar Zowghi. Users' voice and service selection: An empirical study. In *EmpiRE'14*, pages 76–79, 2014.

5. Muneera Bano, Didar Zowghi, Naveed Ikram, and Mahmood Niazi. What makes service oriented requirements engineering challenging? A qualitative study. *IET Software*, 8(4):154–160, 2014.
6. Agustin Casamayor, Daniela Godoy, and Marcelo Campo. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *IST*, 52:436–445, April 2010.
7. Jane Cleland-Huang, Orlena CZ Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. Software traceability: trends and future directions. In *FSE'14*, pages 55–69. ACM, 2014.
8. Johan Natt och Dag, Vincenzo Gervasi, Sjaak Brinkkemper, and Bjorn Regnell. A linguistic-engineering approach to large-scale requirements management. *IEEE Software*, 22:32–39, Janaury 2005.
9. A. Ferrari, G. Lipari, S. Gnesi, and G.O. Spagnolo. Pragmatic ambiguity detection in natural language requirements. In *AIRE'14*, pages 1–8, Aug 2014.
10. Alessio Ferrari and Stefania Gnesi. Using collective intelligence to detect pragmatic ambiguities. In *RE'12*, pages 191–200. IEEE, 2012.
11. Vincenzo Gervasi and Didar Zowghi. Supporting traceability through affinity mining. In *RE'14*, pages 143–152. IEEE, 2014.
12. Rosane S Huergo, Paulo F Pires, Flavia C Delicato, Bruno Costa, Everton Cavalcante, and Thais Batista. A systematic survey of service identification methods. *SOCA*, 8(3):199–219, 2014.
13. Pooyan Jamshidi, Sedigheh Khoshnevis, R Teimourzadegan, Ali Nikravesh, and Fereidoon Shams. Toward automatic transformation of enterprise business model to service model. In *PESOS'09*, pages 70–74. IEEE Computer Society, 2009.
14. Artemios Kontogogos and Paris Avgeriou. An overview of software engineering approaches to service oriented architectures in various fields. In *WETICE'09*, pages 254–259. IEEE, 2009.
15. Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press, 2008.
16. Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: a research roadmap. *IJCIS*, 17(02):223–255, 2008.
17. Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.
18. Robert B. Settle and Linda L. Golden. Consumer perceptions: Overchoice in the market place. *Advances in Consumer Research*, 1:29–37, 1975.
19. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.
20. Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res.*, 37:141–188, 2010.
21. Tassio Vale, Gustavo Bittencourt Figueiredo, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. A study on service identification methods for software product lines. In *SPLC'12, vol. 2*, pages 156–163. ACM, 2012.
22. Martin Wirsing and Matthias Hölzl. *Rigorous Software Engineering for Service-oriented Systems: Results of the SENSORIA Project on Software Engineering for Service-oriented Computing, vol. 6582*. Springer Science & Business Media, 2011.
23. Konstantinos Zachos, Neil Maiden, and Rhydian Howells-Morris. Discovering web services to improve requirements specifications: does it help? In *REFSQ'08*, pages 168–182. Springer, 2008.
24. Ali Taei Zadeh, Muriati Mukhtar, Shahnorbanun Sahran, and MR Khabbazi. A systematic input selection for service identification in smes. *Journal of Applied Sciences*, 12(12):1232–1244, 2012.