Contextualized Semantic Analysis of Web Services

Scott Grant
School of Computing
Queen's University
Kingston, Canada
scott@cs.queensu.ca

Douglas Martin
School of Computing
Queen's University
Kingston, Canada
doug@cs.queensu.ca

James R. Cordy
School of Computing
Queen's University
Kingston, Canada
cordy@cs.queensu.ca

David B. Skillicorn
School of Computing
Queen's University
Kingston, Canada
skill@cs.queensu.ca

***Abstract*—The poor locality of operation descriptions expressed in the Web Service Description Language (WSDL) makes them difficult to analyze and compare in web service discovery tasks. This problem has led us to develop a new method for service operation comparison involving contextualizing operation descriptions by inlining related type information from other sections of the service description. In this paper, we show that this contextualization of web service descriptions can enable topic models (statistical techniques for identifying relationships) to produce semantically meaningful results that can be used to reverse engineer service-oriented web systems and automatically identify related web service operations. Specifically, we model contextualized WSDL service operations using Latent Dirichlet Allocation, and show how this approach can be used to more accurately find similar web service operations.**

***Keywords*-web services, reverse engineering, topic models**

I. Introduction

Web services are software components used to communicate over a network. These web services are often described using domain-specific languages, outlining the operations that are available, the type of messages that can be sent, and other information about the provider.

The structure of service descriptions written in the Web Service Description Language (WSDL), one such domain-specific language, makes reading and understanding them a difficult task. This problem makes it even more difficult to discover relationships between service operations when considering a large repository of web services. Latent topic models can be used to find these relationships, but without adaptation to the specifics of WSDL, they can produce irrelevant noise. Due to the sparsity of local syntax in WSDL operation descriptions, there are not enough tokens in them to support any meaningful semantic conclusions.

In this paper, we use a strategy for restructuring WSDL documents into a set of contextualized operations in an empirical analysis of a repository of web services showing that, by using these contextualized operations, we can find more meaningful relationships when performing Latent Dirichlet Allocation. We use a similarity metric to identify related web service operations based on the derived model, and show by example how significant improvements are made through contextualizing.

The main contributions of this paper are:

- *A description of contextualized web service operations, and why they enable topic models to perform meaningful automated web service discovery.* Although <operation> elements in WSDL are not natively good input source for these techniques due to the small number of tokens they contain, contextualizing them by inserting the other elements they reference can give a topic model an appropriate amount of information to derive these conceptual relationships.
- *An examination of both the global and local improvements that are made after contextualization.* We use a visualization to directly observe the global structure obtained after the operations are contextualized. We identify the main relationships directly from the visualization. We also view the most similar web services before and after contextualization, and show the improvements we are able to make.

II. Motivation

In our previous work on web service analysis with WSDL [13], we attempted to discover similar web service operations using clone detection. We discovered that the organization of these descriptions makes it impossible to identify meaningful operation descriptions to compare. In general, analysis techniques assume that related aspects are grouped together; however, this is not the case with WSDL. Using whole WSDL descriptions is a viable option, but does not yield the desired granularity. We may be able to tell which services are related, but how they are related may remain unclear. To compare operations, we could extract the <operation> elements, but these contain very little information other than the operation name and ignore other valuable information including type definitions. This led us to develop a method to gather the relevant pieces of the operation and consolidate them into self-contained operation descriptions we call WSCells (pronounced "wizzles"). Our previous work demonstrated that WSCells yield much more meaningful results for clone detection. In this paper we experiment with whether they yield a similar improvement for semantic analysis of web services using LDA.

Topic models and other information retrieval techniques like Latent Semantic Indexing [5] have been used to automatically infer semantic relationships based on syntactic information. They are built on the idea that the tokens used in a data set contain enough information to extract semantic relationships. In order to analyze web service descriptions, we would like

to discover these relationships in WSDL operations. However, due to the sparsity of local syntax in WSDL operation descriptions, there are not enough tokens in them to support any meaningful semantic conclusions. For these reasons, we believe that utilizing concept location techniques as a way to identify the relationships between individual web services is an ideal way to demonstrate that WSCells add important semantic structure.

III. Background

A. Web Service Description Language (WSDL)

A WSDL description of a web service contains an interface definition of one or more operations provided by that service. However, each definition is broken up into pieces depending on what aspect they define and linked together in a chain. The pieces are organized into 5 main sections:

1) Types: The types section contains type definitions for exchanging data between a client and a service using the XML Schema Language (XSL). The schema may define complexTypes, which are essentially objects that contain other elements, or simpleTypes, which may be restrictions of primitive types, enumerated types, or patterns (among others). Other elements may be declared as these types, thereby inheriting the elements contained within.

2) Messages: Messages define the data elements corresponding to the input, output and faults of each operation. They contain one or more parts, which may refer to elements defined in the types section. These parts can represent parameters of the operation, but often they simply refer to an element in the types section that contains the parameters.

3) Port Types: The port types section contains one or more operations that make up the web service. Each of these operations may contain an <input> or <output> element depending on what kind of communication takes place (e.g. request-response, notification, etc.). It may also contain any number of faults. These inputs, outputs and faults refer to messages defined elsewhere in the file.

4) Bindings: Bindings define a message format and protocol for a port type. Often this protocol is Simple Object Access Protocol (SOAP).

5) Services: The services section defines a group of ports, which define an endpoint and specify an address for a binding.

For the purposes of similarity analysis, we ignore the Bindings and Services sections because they contain information specific to the service and are not likely to contain anything useful for identifying relationships.

The description of an operation begins in the <portTypes> element where operations are listed in their own <operation> element. Each of these elements contain a number of <input>, <output> or <fault> elements that correspond to a <message> element defined somewhere else in the description. These contain <part> elements that may refer to other remote elements in the <types> element. The elements in the <types> element can also contain elements that have other types associated with them, which can contain more elements, and so on. The result is that operation descriptions

```
<source file="HotelReservation.wsdl" startline="81" endline="85">
 <operation name="ReserveRoom">
  <input message="ReserveRoomRequest">
   <message name="ReserveRoomRequest">
    <part name="body" element="ReserveRoomRequest">
     <element name="ReserveRoomRequest">
      <element name="payment" type="Payment"/>
       <element name="ccNumber" type="int"/>
       <element name="cardHolder" type="string"/>
       <element name="expiryDate" type="date"/>
      </element>
      <element name="room" type="Room">
       <element name="roomID" type="int"/>
       <element name="numBeds" type="int"/>
       <element name="isSmoking" type="boolean"/>
      </element>
     </element>
    </part>
   </message>
  </input>
  <output message="ReserveRoomResponse">
   <message name="ReserveRoomResponse">
    <part name="body" element="ReserveRoomResponse">
     <element name="ReserveRoomResponse"/>
    </part>
   </message>
  </output>
  <fault message="RoomNotAvailableException">
   <message name="RoomNotAvailableException">
    <part name="body" element="RoomNotAvailableException">
     <element name="RoomNotAvailableException"/>
    </part>
   </message>
  </fault>
 </operation>
</source>
```

Fig. 1. An example WSCell for a "ReserveRoom" operation of a simple hotel reservation service, which takes a "Payment" element and a "Room" element as input and sends an acknowledgment back to the client. Simple <input>, <output> and <fault> elements are expanded to include the <message> elements to which they refer. Then, each <part> element of each message is expanded with the corresponding element in the types section. Finally, each element with a complexType is expanded to include its type definition. This continues recursively until only primitive types remain.

are split into remote pieces and intermingled in the same description. This poses a problem for analysis techniques, such as LDA, because operations can not be easily separated into units.

B. Latent Dirichlet Allocation (LDA)

A topic model is a statistical model used to identify a set of unobservable latent topics in a data set. The fundamental premise behind a topic model is that there is some correlation observed among the tokens in the data set that can be explained by a mathematical relationship between them, and that these relationships can be extracted as topics. One example of a topic model is Latent Dirichlet Allocation (LDA) [3], a generative model that assumes the data set was derived from a prior topic distribution over the data. In general terms, LDA assumes the existence of a number of topics that can be used to relate elements of the data set to one another. If two pieces of data are strongly related to the same topic (or a similar set of topics), they are likely to be very similar to one another.

```
stockquote (41)
<wsdl:operation name="GetQuote" >
  <wsdl:input message="tns:GetQuoteSoapIn"/>
  <wsdl:output message="tns:GetQuoteSoapOut"/>
</wsdl:operation>

WSAmazonBox (764)
<wsdl:operation name="AuthorBox" >
  <wsdl:input message="tns:AuthorBoxSoapIn"/>
  <wsdl:output message="tns:AuthorBoxSoapOut"/>
</wsdl:operation>
```

Fig. 2. Two unrelated web services that are incorrectly found to be similar when analysed without contextualization, along with the web service provider and line number where the operation was found. The stock quote operation and author operation share a SOAP interface, but are otherwise in completely different domains. The SOAP keyword is identified by the topic model with no contextualization, and treated as important enough to link these two operation together. With contextualization, this problem is alleviated.

In LDA, a common distance metric involves taking the Hellinger distance between the probability distributions for each pair of documents [2]. The Hellinger distance is a similarity metric that, in this context, helps to find pieces of data that have similar topic relationships. For example, in a simple latent topic model with two topics, each document itself is represented as a distribution over them, or essentially as a pair of probabilities. If two pieces of data have very similar probabilities of being related to the topics, those pieces of data would have a very low Hellinger distance, and therefore a very high likelihood of being similar. When the overall Hellinger distance data is normalized to a value in the range $0 < x < 1$, and then subtracted from 1 (we want low distance scores to correspond to high probability of relationship, so distance 0 should be similarity 1), each individual distance score provides the means to estimate the probability that two documents are conceptually related to one another.

The following table gives the probability distribution for each of the documents for two sample topics t_1 and t_2, and the Hellinger distance from each other document (larger numbers indicate less similarity).

	t_1	t_2	$h(d_1)$	$h(d_2)$	$h(d_3)$
d_1	0.33	0.67	-	0.003	0.258
d_2	0.40	0.60	0.003	-	0.210
d_3	0.95	0.05	0.258	0.210	-

By using LDA as a model and calculating the Hellinger distance between the topic distributions, it can be seen that d_1 and d_2 are the most similar documents. After normalizing the data, smaller distances like the one between d_1 and d_2 will be very close to 1, and larger distances like the one between d_1 and d_3 will end up near 0.

IV. Method

A. Approach

In our previous work, we looked for a way to leverage existing code clone detection tools to find similarities in a web

Before contextualization:

```
<wsdl:operation name="GetBRAZIBOR" >
  <wsdl:input message="tns:GetBRAZIBORSoapIn"/>
  <wsdl:output message="tns:GetBRAZIBORSoapOut"/>
</wsdl:operation>
```

After contextualization:

```
<operation name="GetBRAZIBOR" >
  <input message="tns:GetBRAZIBORSoapIn">
...
    <element name="BRAZIBORTypes">
      <s:restriction base="s:string">
        <s:enumeration value="Overnight"/>
        <s:enumeration value="OneYear"/>
...
  <output message="tns:GetBRAZIBORSoapOut">
...
    <element name="Currency" type="tns:Currencies">
      <element name="Currencies">
        <s:restriction base="s:string">
          <s:enumeration value="USD"/>
          <s:enumeration value="AED"/>
          <s:enumeration value="AFA"/>
...
    <element name="Date" type="s:string"/>
    <element name="Value" type="s:double"/>
    <element name="Text" type="s:string"/>
    <element name="Source" type="s:string"/>
    <element name="Description"
             type="tns:RateDescription">
      <element name="Type" type="tns:RateTypes">
        <element name="RateTypes">
          <s:restriction base="s:string">
            <s:enumeration value="FederalFunds"/>
            <s:enumeration
                value="FederalFundsTargetRate"/>
            <s:enumeration value="Libor1Month"/>
            <s:enumeration value="Libor2Month"/>
...
    <element name="Description" type="s:string"/>
    <element name="Name" type="s:string"/>
    <element name="Maturity" type="s:string"/>
    <element name="MaturityUnit"
             type="tns:MaturityUnitTypes">
      <element name="MaturityUnitTypes">
        <s:restriction base="s:string">
          <s:enumeration value="Week"/>
          <s:enumeration value="Month"/>
          <s:enumeration value="Year"/>
          <s:enumeration value="Day"/>
...
    <element name="MaturityCount" type="s:int"/>
    <element name="SeasonallyAdjusted"
        type="s:boolean"/>
    <element name="Availability" type="s:string"/>
    <element name="Source" type="s:string"/>
    <element name="Discontinued" type="s:boolean"/>
    <element name="Service" type="s:string"/>
    <element name="Suffix" type="s:string"/>
    <element name="Factor" type="s:int"/>
    <element name="Precision" type="s:int"/>
...
```

Fig. 3. An example of a web service that is particularly well suited to contextualization. The basic operation holds almost no information about the type of data it is designed to handle. After contextualizing it, the operation contains a full set of currency data, similar rate types, and other information about how it can be configured.

service repository. We found that the poor locality of WSDL operation descriptions made it difficult to extract a set of potential clones for comparison. Clone detectors assume that related blocks of code are grouped together in a continuous sequence of lines; however, this is not true for WSDL where <operation> elements will contain references to elements in other parts of the description.

To solve this problem, we developed a method to give context to these bare operations by injecting the referenced elements into the operation itself. We call these contextualized operations Web Service Cells, or WSCells, because they are like the cells that make up a web service. To do this, we used a source transformation language called TXL [4]. The TXL program takes a single WSDL document, extracts the base operation (the <operation> element), and inserts the referenced elements into the element that references it. So for an <input> inside the operation, the corresponding <message> is found and inserted into it; for each <part> inside the <message>, the corresponding <element> is found and inserted; and so on, until there are no more elements left. For example, consider the WSCell, shown in Figure 1, for a "ReserveRoom" operation of a simple hotel reservation service. This operation takes a "Payment" object and a "Room" object as input and sends an acknowledgement or a fault in return. The WSCell includes all referenced elements from each section of the WSDL description inserted into the elements that reference them.

Our research demonstrated that WSCells showed a clear improvement in the identification of similar operations when using clone detection. It convinced us that the contextualization of WSDL operation descriptions can make it easier to find related operations using existing tools. With that in mind, we take a similar approach and apply it to topic models to show that conceptual relationships can be identified in a similar way.

B. Goals

We performed a comparison of the bare <operation> elements and the contextualized operations (WSCells) to see how well LDA was able to model meaningful relationships between them. In each case, we generated a model of the data, and for each WSDL operation, used the Hellinger similarity metric described in Section III-B to identify the other most similar operations. We then examined the list of the most similar operations for each individual web service to see if contextualizing provided a better set of related web services. We evaluated several topic counts empirically to determine an appropriate value. For this data set, we used a hard value of 100 topics.

Our experimental data is a set of WSDL service description files with over 500 service descriptions containing over 7,500 operations from a wide variety of domains, obtained through a web services search engine by Seekda [1]. Our goal is to show that contextualizing web service operations provides a clear improvement in the ability of a topic model to identify related web services. We will use a global analysis to visualize overall structural improvements and large-scale sets of related features, and a local analysis to directly observe how actual recommendations can be improved for individual web service operations.

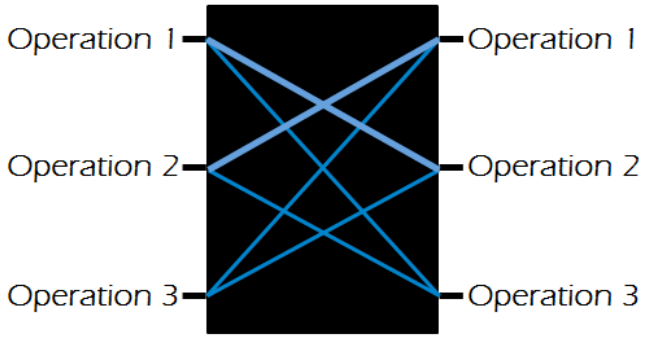

Fig. 4. A visual representation of how Bluevis maps functions to positions on either side of the screen. Strong conceptual relationships are marked by a strong line. For comparison, Figure 5 shows two examples of the Bluevis display with real web service operations. It is also important to note that any self-references are removed, so the display is not dominated by horizontal lines.

V. Analysis

A. Global Structure

The global view is based on a visualization tool we have developed called Bluevis. Bluevis explores how the overall file structure of the system relates to its conceptual distribution. Figure 4 provides a simple abstraction of the visualization to show how related web services are linked together. If a line extends from one side of the window to the other, it represents a conceptual relationship (above a certain threshold) between those two web services. Larger blocks of related web services form brighter lines, and random pairings appear as dark and almost invisible. In this way, the global conceptual structure of the system can be seen. The list has operations from the same service in a similar area, but is otherwise unordered.

We will use Figure 5 to demonstrate the visualization. Each image represents a single model generated from the web service data. On the left, the WSDL operations are used as input to the model. The only preprocessing step that we took in this example was to split apart the camel-case and underscore-separated compound tokens. Without this step, there would be almost no meaningful shared tokens between operations. On the right, the WSCells are used as input to the model, and any compound tokens are also split apart in the same way.

At first glance, the raw WSDL operations on the left provide a much more visually chaotic semantic structure than the WSCells on the right. On closer inspection, this is indeed true. Many of the similar operations identified by using the simple WSDL operations as input are meaningless, and show up due to shared tokens like *get* or *SOAP*. It appears that some clear structure is discovered by the model in both cases. Horizontal lines indicate operations that are likely from the same service or service provider, and diagonal lines indicate potentially similar operations from other sources. It is important to note that the visualization is mirrored horizontally, and any diagonal

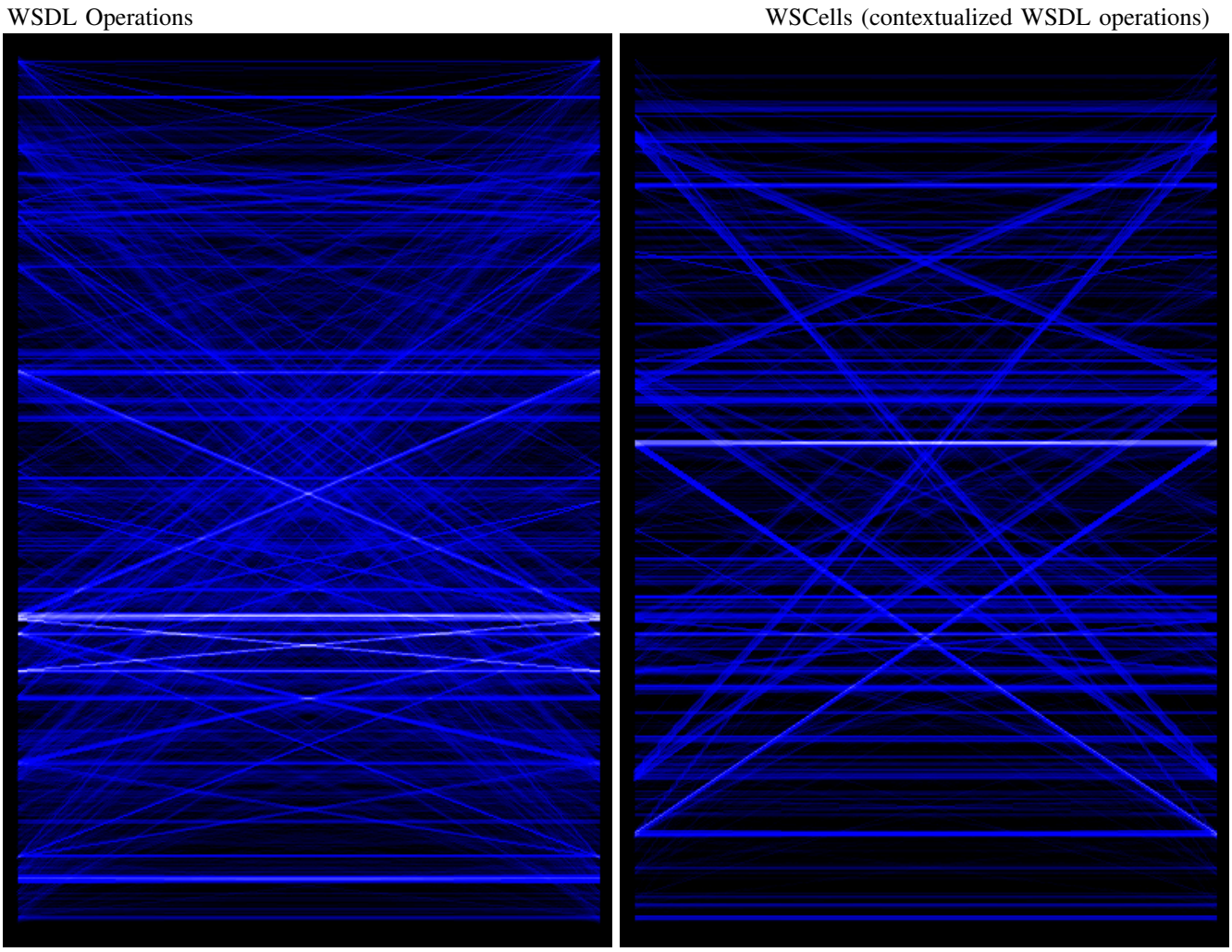

Fig. 5. Two screenshots of Bluevis's interface. The left and right sides of each screenshot correspond to an identically ordered list of the web service operations, and a connection made between the sides represents a conceptual similarity between two different operations above a certain user-defined threshold. In this example, we show the top 25,000 pairs as determined by the similarity metric described in Section III-B. On the left, the basic WSDL operations have been used to generate a model. On the right, WSCells are used. The bare operations on the left show some simple structure, but otherwise show very sparse relationships between relatively unrelated data. The WSCells on the right are able to demonstrably capture larger groups of related web services.

line will show up as an X in the image (any connection between *a* and *b* will also result in a connection from *b* to *a*). Bright white lines appear where large collections of related web services overlap, which can indicate either strong local relationships in a single provider, or a strong indication that a set of web services are cloned by another provider.

The images on their own are not enough to suggest that the appearance of structure is a definitive proof that using contextualized operations allows topic models to identify related web services. To really understand what type of information is being identified, we performed an analysis of the large global features uncovered by this visualization.

We will first examine the operations diagram on the left in detail. Most of the structure is formed by small blue lines that connect across web systems. Due to the splitting of compound tokens, general terms (this may include terms like *load*, *application*, *order*, and *log*) are indications for LDA to treat those operations as related. The majority of these solitary connections are between unrelated operations that happen to share a few tokens in either their name or in the input or output messages. An example can be seen in Figure 2. The two operations in question deal with completely different domains, and are only related due to the shared use of the SOAP interface. Many of the other tokens are common to all operations (*wsdl*, *operation*, *message*, and do not provide any value for the model. This motivates our use of contextualization; with additional context in the form of tokens, a topic model is able to identify more meaningful relationships.

In the center of the left visualization (Figure 5) is a large X that shows a large block of related operations found in two separate sections of the list. Although these are not strict clones of each other, they are related by subject area, they are offered by the same web service (although through a separate

interface), and that they all include the “parameterOrder” element to provide additional context. The parameter list has a greater influence on the results of the model due to the introduction of many new tokens. Many elements of a WSDL operation are optional, and without contextualization, these rare instances stand out greatly amidst other bare operations.

At least two more groups of related web services are found below the large X previously discussed, and the one immediately below stands out as being particularly important in the model. The smaller but brighter X immediately below is a collection of web services from the same provider; the upper block is entirely related to orders, and the bottom block is entirely related to users. The crossover between the blocks is due to the inclusion of fault tags in the operations. Relatively few of the operations include this information, so like the parameter list in the previous example, these tokens are a strong indication of similarity.

While it is true that the inclusion of parameter lists and fault tags can be an indication of similarity, these data are optional, and not commonly used in the real examples we examined. The naive approach of using basic operations with topic models gives some context about the web services, but in general, the results are either trivial duplication of common tokens, or web services that are found through the same provider. However, the goal of web service discovery across systems using topic models is made more achievable through contextualization.

The visualization on the right side of the figure, using contextualized operations as input to LDA, shows a significant reduction in the number of sparse random connections. When we investigate these closer, it appears that the majority of the information shown by the visualization contains a relevant semantic structure. Several large fan-out points appear, indicating web services that have similar operations offered by other providers. Large horizontal blocks indicate clusters of related operations from the same web service. To better understand what the hot points in the visualization actually indicate, we take a closer look.

One large set of related operations connected by diagonal lines can be seen starting near the top, running down to the bottom, and finally joining just above the center point of the image. These services were detected by the previous model, and a close examination of the left image will show some faint lines running between those areas. However, they do not stand out amidst the other functions that appear to be related, and may be easily missed. When we examined these, it turned out that they were three related collections of operations to retrieve holiday dates. Each collection was for a different geographical region, and included operations like GetHalloween, GetBoxingDay, and GetGuyFawkesNight. Even with these terms being duplicated across the three collections, the naive approach to using raw WSDL operations did not uncover this association.

Another large section of related operations can be seen in the large X in the bottom half of the visualization. The majority of these are found in a cluster of operations that

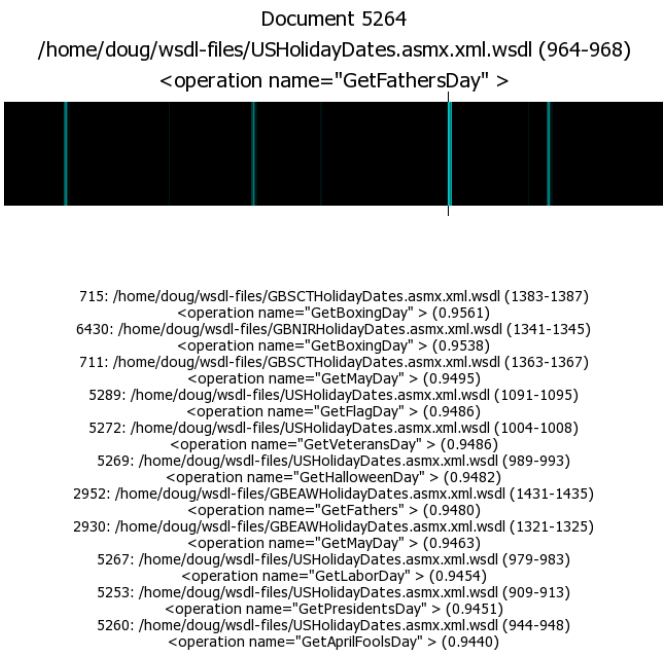

Fig. 6. We used a visualization tool designed to show how topic models actually relate individual web services by conceptual similarity. The uppermost part of the display gives information about the currently selected web service, including the WSDL file where it is found. Immediately below that is a heatmap corresponding to the list of web services found vertically along each side of Bluevis, with blue blocks indicating the presence of related web services. Next is an optional description of the code fragment that the mouse is hovering over in the heatmap, if desired, to examine related blocks that emerge in the heatmap. Below that is a list of the top related web services.

provide bank reference rates for lending. They have names like GetEURIBOR, GetBRAZIBOR, and GetMOSIBID, and intuitively share similarities. However, the operations are very terse, and the input and output messages simply append SoapIn and SoapOut, as in the earlier examples from Figure 2. After contextualization, a portion of the new WSCell can be seen in Figure 3. A great deal of additional information about the applicable currencies, related rate types, and other financial information is explicitly added to the description. These keywords are likely to be shared with other financial web services, and a topic model like LDA will be able to use these tokens to detect similarities between them.

Although the majority of related operations in this block are calls to get specific related rates (GetSIBOR, GetLIBOR, and so on), there are additional relationships detected from other areas of the collection. This indicates that LDA is able to identify web services from other providers when the topic model is able to draw on the explicit data added by contextualization; LDA is not able to make these connections with the raw WSDL operations.

B. Local Similarity

To really demonstrate the value of contextualization for use with topic models, we examine the similarity between individual operations using the similarity metric discussed in Section III-B. By using the Hellinger similarity as defined, the

entire set of operations can be ranked and ordered from most relevant to least relevant. This allows us to compare both the standard WSDL operations and the WSCells to evaluate which approach garners more meaningful results.

In Figure 6, a screenshot of our visualization tool POCO (Pairwise Observation of Concepts) is shown. It uses a topic model to identify the pairwise relationships between individual operations, and provides a heatmap overview along with a list of the other most similar operations. In this study, we focused on the list of the top related operations, with the assumption that a user who was interested in identifying related operations would want to see this data.

If we look even more closely at Figure 6, the top results for a single operation (GetFathersDay, from the USHolidayDates service) can be directly observed. This particular method is clustered in among the holiday web services previously discussed in Section V-A. Even the clusters along the sides at the endpoints of the X can be seen in this view, where several dense areas of related operations that deal with holidays in different services are identified by the model. These operations all share a related naming convention, which would intuitively lead to the assumption that the bare WSDL operations would also be able to identify these related methods. In fact, when examining the top results for the same operation without contextualization, the list includes GetLogo and GetEURIBOR, two web services with no meaningful relationship to GetFathersDay. With a small number of meaningful tokens, the non-contextualized results simply do not have enough context to allow LDA to form meaningful conceptual structure, and the results can not be used for web service discovery in the way that WSCells can.

Figure 7 provides a side-by-side examination of the results of contextualization for a single operation. In this example, the GetCEO operation from the xinsider service was used. The data on the left are the most related operations for non-contextualized WSDL operations, and the data on the right are the most related operations for contextualized WSCells.

To show that this type of improvement is common, we expand our view to look at the most similar operation for other examples. In Figure 7, we show how the set of most similar operations is also greatly improved. This is an expansion of the kind of data we see from Figure 8. In some cases, such as the GetReservations operation, the improvement is clear. With non-contextualized operations, LDA suggests that the most similar operation is GetSOFIBOR. With WSCells, LDA instead suggests GetRoomAvailabilityForDay. Some other cases are not as clear, as with the GetIssueData operation. The non-contextualized suggestion is word_cloud, and the WSCells suggestion is GetFlightData. In the majority of cases, when compared side-by-side like this, the contextualized operations tend to be more useful in a human-oriented context.

The data from Figure 7 helps to demonstrate that this approach can be used for identifying specific semantically related operations. We address a concern with similar operations being found in the same service in the contextualized example as a threat to validity, and note that filtering out operations from the same service still results in markedly better results than bare WSDL operations alone.

(a) Similar WSDL Operations

Operation Name	Service Name
getDVDShops	seawiseservice (1340)
Book	K4TAirSell (240)
GetWebsites	KYWOrgData (584)
GetWeatherReport	usweather (55)
GetCEO	xinsider (1603)
GetEaster	USHolidayDates (934)
GetSports	livescoresservice (7293)
GetTURKIBOR	xinterbanks (5993)
GetBookTitles	BibleWebService (192)
GetData	DataParam (296)

(b) Similar WSCells

Operation Name	Service Name
GetIssuerOwnerships	xinsider (1700)
GetDirectors	xinsider (1593)
GetDirectors	xinsider (1685)
GetInsiderTransactions	xtibco (2474)
GetInsiders	xtibco (2387)
GetOfficerCompensations	xcompensation (509)
GetIssuerTransactions	xinsider (1643)
GetIssuerOwnerships	xinsider (1608)
GetIssuerOwnerships	xinsider (1710)
GetRoster	xinsider (1680)

Fig. 7. An examination of the results of contextualization for a single operation. In this example, the GetCEO operation from the xinsider web service was used. The upper table (a) shows the most related operations as discovered by the topic model with no contextualization. Only one of the operations seems plausible at first glance (the alternate GetCEO operation in xinsider, starting at line 1603). The lower table (b) shows the most related operations for GetCEO as discovered by the topic model with contextualization. These results seem much more useful, and still manage to span several related web services.

VI. Summary and Future Work

By contextualizing WSDL operations and generating a set of WSCells to use as input for an LDA model, we have demonstrated that the results after contextualizing provide a significant improvement in web service discovery using topic models. The sparsity of tokens in high-level languages such as WSDL do not provide enough context for a topic model to make meaningful conclusions.

With a global view of the web services, we were able to examine large-scale structure identified by LDA for both models. In the case of the bare WSDL operations, we were able to see a great deal of random scattering with little value, and a large block of related web services of questionable value. With WSCells, we see a greater amount of cross-system relationships, and far less random connections between uninteresting operations. These results can be observed globally using visualizations, and examined in detail by using a pairwise similarity metric.

Operation	Most Similar WSCell	Most Similar WSDL Operation
ListFinancials xfinancials (2508)	**GetFinancialServicesItemList** xfinancials (2548)	**LanguagesList** Articles (432)
ExportShipsAndCategories Export (319)	**ExportItineraryAndSteps** Export (324)	**Search** xscreener (1055)
GetIssueData xemerging (718)	**GetFlightData** FastTrack (438)	**word_cloud** taporwareServices (434)
GetWeatherReport usweather (41)	**GetWeather** globalweather (76)	**GetIndices** xquotes (1813)
GetAIDIBOR xinterbanks (5455)	**GetTRLIBOR** xinterbanks (5425)	**GetCarriers** blackbox (9303)
searchByIdentifier icontest (1240)	**searchByNameAndAddress** icontest (1225)	**GetLastSecurityHeadlines** xreleases (594)
ToolsAndHardwareBox WSAmazonBox (734)	**KitchenAndHousewareBox** WSAmazonBox (699)	**ListRenditions** DocMan (1350)
GetReservations holidayguide (467)	**GetRoomAvailabilityForDay** holidayguide (455)	**GetSOFIBOR** xinterbanks (5597)
GetOtherProductInfo blackbox (9207)	**NextOtherProductPortion** blackbox(9203)	**GetParkingInfo** blackbox (9251)
GetAllSplitsByExchange xglobalhistorical (1836)	**GetAllCashDividendsByExchange** xglobalhistorial (1851)	**GetTeamLoyalties2** livescoresservice (6749)

Fig. 8. A side-by-side comparison of the most relevant operation as determined by LDA for the bare WSDL operations and the contextualized operations. The left column gives the operation, the service (WSDL file), and the starting line number, given to remove ambiguity if two web services share the same name. In each of these examples, we believe WSCells are able to provide results that are more useful. These ten examples are typical of the majority of operations affected by contextualizing. For an example of the improvement for a single operation, see Figure 7.

A local view of the improvements helps to explicitly demonstrate how individual operations can be related to one another, and specifically how the relationships become more useful after contextualization. With the added information about the services, a topic model like LDA is able to use the new tokens to derive meaningful correlations between operations.

In future work, we would like to examine how well the results generated from topic models actually correspond to human-oriented opinion. For example, we would like to see if the best results for a given operation, such as the GetCEO method used in Figure 7, are actually similar to the best matches that a human would hope to find. Our previous experience with software clones and topic models has given us some experience in evaluating the performance of these techniques [6]. With an evaluation from users on the quality of the results, we would be able to quantify the improvement over models generated from bare WSDL operations. We would also like to use this approach with other XML-based modelling languages to see if the same benefits can be obtained.

VII. Threats to Validity

One significant concern with the Bluevis visualization is the fact that the web services are not ordered in any significantly meaningful way. In our data set, operations offered by the same service are grouped together locally. This often leads to visual structure in the diagram due to the strong relationship between all of the operations within the same service due to shared data types. The X shape seen by clusters of related operations in different areas of the list does not immediately indicate why the group on the top or the group on the bottom are in that specific location. Rather than focus on the order of the documents, we have attempted to look at two specific features brought out in this view. First, the reduction in "noise" across the entire set of relationships when using a contextualized data set appears to show fewer meaningless relationships, such as two operations that both get something using SOAP, as seen in Figure 2. Second, the large blocks of structure can be examined in-place in the visualization. This allows us to actually identify the most significant features discovered by LDA, and to directly evaluate how accurate the relationships are from a human-oriented standpoint.

In many cases, contextualizing WSDL operations has resulted in similarity lists that are localized to the same web service. For example, in Figure 7, it can be argued that the results are not useful because the majority of operations come from the same xinsider service. We can mitigate this issue by noting that these data can easily be filtered out, and that the remaining results across other web services are still very relevant. From our observations, web service providers are likely to use common terms across their own web services. The important descriptive terms that allow a topic model to identify context are still shared across providers.

WSDL is only one example of a language for describing web services, and it may be argued that this technique will not apply to a general case. A strong benefit of this approach is that it applies very well to XML-based modelling languages, where the attributes tend to be scattered over the entire description instead of localized to the web service operations. We believe

that contextualizing web service descriptions will allow topic models to identify context in a wide range of modelling languages, and see no indication that this approach is restricted to WSDL.

VIII. Related Work

With the growth of the web, along with the corresponding growth of web services, a concerted effort has been made to provide automated web service discovery available. Paolucci et al. [15] discussed the claim that a semantic representation of web services would enable matching for related web service capabilities. In particular, they identified WSDL's lack of semantic information for this specific problem, and proposed DAML-S, a language for service description based on the DARPA Agent Markup Language (DAML). The authors suggested that an XML-based standard for web service description lacked sufficient semantic information, and that "two identical XML descriptions may mean very different things depending on the context of their use." This observation remains true today, and was a motivating factor in our decision to use contextualization. The goals of the semantic web continued to evolve towards rich semantic specifications for web services and operations [9], [12].

The first approach to using information retrieval techniques for web service discovery was in 2005, when Platzer and Dustdar used WSDL files in conjunction with the vector space model [16]. The authors used the vector space model to tokenize each web service description and to compare them against each other using the cosine similarity, a common similarity metric. Although WSDL files can be sparse, they discovered that a even a basic analysis of the keywords found in WSDL sources can be enough to build a usable search engine for web service discovery based on their approach. Paliwal et al. [14] extended this idea with a novel approach to web service discovery by applying Latent Semantic Indexing (LSI) to information derived from WSDL service descriptions. In their work, they also faced the issue of limited information in the service descriptions, and addressed the problem by linking results with a domain ontology. Our shared solution goal was the addition of implicit semantic information, and through the use of an ontology, the implicit data becomes explicit, allowing models like LSI and LDA to find good relationships.

Ma et al. [11] also developed an approach for web services discovery and an initial divide and conquer strategy followed by a singular value decomposition. To address the ever-present issue of a lack of context in web service description files, the authors clustered sets of operations together to discover a relevant group of services. Once they found a clustered group that was most similar to their desired query, they then applied a matrix decomposition that is also used with LSI to identify the most similar web service. With their approach, the authors aim to eliminate as much of the irrelevant services from the problem set to give their algorithm a better chance of finding an appropriate solution. The authors also experimented with Probabilistic Latent Semantic Indexing [10], a technique that LDA is based on, but they do not provide a great deal of data to evaluate.

In a recent survey of survey discovery approaches, Rambold et al. performed a comparison of 42 different approaches. Of the techniques evaluated, only a handful dealt with related approaches like the vector space model or LSI. In Kokash et al. [7], a combination of lexical and structural matching techniques are used to evaluate the similarity between concepts. Lee et al. [8] also build a vector space model representation of the web services using a clever system for encoding service information into elements of an ordered tree, and execute similarity requests through SQL queries to a standard database.

IX. Conclusions

Topic models are an automated way for effectively identifying conceptually related pieces of data. Although web service languages are not natively good input sources for these techniques due to their sparse descriptions, contextualizing the operations can give a topic model an ideal amount of information to derive these conceptual relationships.

Contextualizing has already proven to be useful with clones, and we have demonstrated how it can be used to effectively analyse a set of web systems and to discover related web services.

Acknowledgements

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada, by the Ontario Graduate Scholarship Program, and by an IBM Center for Advanced Studies Fellowship.

References

[1] "Web Services Search Engine," http://webservices.seekda.com.

[2] D. M. Blei and J. D. Lafferty, "A correlated topic model of science," *Annals of Applied Statistics*, vol. 1, pp. 17–35., 2007.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[4] J. R. Cordy, "The TXL source transformation language," *Science of Computer Programming*, vol. 61, pp. 190–210, August 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1149670.1149672

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[6] S. Grant and J. R. Cordy, "Vector space analysis of software clones," in *17th IEEE International Conference on Program Comprehension (ICPC '09)*, May 2009, pp. 233–237.

[7] N. Kokash, W.-J. van den Heuvel, and V. DAndrea, "Leveraging web services discovery with customizable hybrid matching," in *Service-Oriented Computing ICSOC 2006*, ser. Lecture Notes in Computer Science, A. Dan and W. Lamersdorf, Eds. Springer Berlin / Heidelberg, 2006, vol. 4294, pp. 522–528.

[8] K.-H. Lee, R. Miguez, S. Serrano, L. Anido, and M. Llamas, "To maximize web service retrieval," in *Proceedings of the 2007 International Conference on Convergence Information Technology*, ser. ICCIT '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 2318–2325.

[9] L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology," in *Proceedings of the 12th international conference on World Wide Web*, ser. WWW '03. New York, NY, USA: ACM, 2003, pp. 331–339.

[10] J. Ma, J. Cao, and Y. Zhang, "A probabilistic semantic approach for discovering web services," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 1221–1222.

[11] J. Ma, Y. Zhang, and J. He, "Web services discovery based on latent semantic approach," in *Proceedings of the 2008 IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 740–747.

[12] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, "Bringing semantics to web services: The owl-s approach," in *Semantic Web Services and Web Process Composition*, ser. Lecture Notes in Computer Science, J. Cardoso and A. Sheth, Eds. Springer Berlin / Heidelberg, 2005, vol. 3387, pp. 26–42.

[13] D. Martin and J. R. Cordy, "Towards web services tagging by similarity detection." in *The Smart Internet*, M. Chignell, J. R. Cordy, J. Ng, and Y. Yesha, Eds., 2010, pp. 216–233.

[14] A. V. Paliwal, N. R. Adam, and C. Bornhovd, "Web service discovery: Adding semantics through service request expansion and latent semantic indexing," *Services Computing, IEEE International Conference on*, vol. 0, pp. 106–113, 2007.

[15] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *The Semantic Web ISWC 2002*, ser. Lecture Notes in Computer Science, I. Horrocks and J. Hendler, Eds. Springer Berlin / Heidelberg, 2002, vol. 2342, pp. 333–347.

[16] C. Platzer and S. Dustdar, "A vector space search engine for web services," in *Proceedings of the Third European Conference on Web Services*, ser. ECOWS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 62–.