An Open Vocabulary Semantic Parser for End-User Programming using Natural Language

DOI:
10.1109/ICSC.2018.00020

Document Version
Accepted author manuscript

Link to publication record in Manchester Research Explorer

Citation for published version (APA):

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
An Open Vocabulary Semantic Parser for End-User Programming using Natural Language

Juliano Efson Sales
Department of Computer Science and Mathematics
University of Passau
Passau, Germany

André Freitas
School of Computer Science
The University of Manchester
Manchester, United Kingdom

Siegfried Handschuh
Department of Computer Science and Mathematics
University of Passau
Passau, Germany

Abstract—The ability to automatically interpret natural language commands and actions has the potential of freeing up end-users to interact with software artefacts without the syntactic, vocabulary and formal constraints of a programming language. As most semantic parsers for end-user programming have been operating under a restricted vocabulary setting, it is unclear how these approaches perform over conditions of high semantic heterogeneity (e.g. in an open vocabulary). As the generation of annotated data is costly and time-consuming, models that effectively address complex learning problems constrained under the assumption of small annotated data sets are highly relevant.

In this paper, we propose a semantic parsing approach to map natural language commands to actions from a large and heterogeneous frame set trained under a small set of annotated data. The semantic parsing approach uses the combination of semantic role labelling, distributional semantics geometric features and semantic pivoting in order to address the semantic matching problem in an open vocabulary setting.

I. INTRODUCTION

The application of semantic parsing to support end-user programming (EUP) using natural language has been applied in different domains of discourse, from the automation of operating systems tasks [1] and application controlling [2], to robotic movement control [3] and programming new mobile behaviour[4]. The ability to automatically interpret commands and actions using natural language has the potential of freeing up end-users to interact with software without the syntactic, vocabulary and formal constraints of a programming language.

The majority of semantic parsers targeting the interpretation of natural language commands have focused on the interpretation over small target frame sets, aiming at a particular domain[5], [6], [7]. This is reflected in semantic parsing models which are evaluated under more constrained vocabulary and syntactic (and thus) semantic heterogeneity conditions.

More recently, test collections targeting open-domain/large command sets/large vocabulary/large syntactic heterogeneity for EUP scenarios have emerged [8]. These test collections reflect the existing opportunity of building semantic parsers which could bridge the semantic gap between end-users and the growing availability of software resources (libraries and service hubs such as IFTTT¹ and Mashape²). These software resources are created by different designers/programmers with different conceptualisations, reflecting different contexts of use and requirements. As most semantic parsers for EUP have been operating under a small vocabulary and more coherent discourse assumptions, it is unclear how these approaches generalise over high heterogeneity conditions.

Frequently machine learning techniques are at the centre of current semantic parsing methods[9], [5], [10]. The assumption behind the application of machine learning methods relies on the existence of training data on a scale proportional to the complexity of the task: simple tasks can be addressed with a relatively small data set, while more complex tasks demand large-scale annotated data[11].

As the generation of annotated data is costly and time-consuming, models that effectively address complex problems constrained under the assumption of small annotated data sets are highly relevant.

This paper proposes a semantic parsing approach which targets large and heterogeneous frame sets and operates under the restriction of small annotated data sets. The proposed model consists of a distributional semantic parsing method with a semantic pivoting heuristic. The proposed representation uses geometric features over different distributional semantic spaces to generate alignment hypothesis between natural language terms and frames in an unsupervised manner. The pivoting method consists of a classification approach which uses the support of distributional and induced type features to provide a score of the semantic reliability of the alignment. The final semantic parsing approach aims at a supervised method which can operate over open/multi-domain vocabularies and can generalise from smaller training sets.

This paper is organised as follows: Section II describes similar tasks and research initiatives related to the semantic parsing of natural language commands. In Section III, we formally present the problem of mapping commands to action frames. Sections IV and V describe respectively our proposed method and the experiment setup. In Section VI we analyse the results. Finally, Section VII provides the final considerations and the conclusion.

¹ifttt.com
²mashape.com
II. BACKGROUND

One of the first initiatives to create a test collection for semantic parsing is the Air Travel Information System (ATIS) data set[12] which includes requests about travel information such as querying flights under cities and dates restrictions [13]. The ATIS test collection is a typical narrow-domain semantic parsing task which is covered with a large training set in which traditional machine learning techniques had succeeded [10].

More recently, most EU semantic parsing scenarios have shifted to parsing natural language commands within the context of robotics. The Human Robot Interaction Corpus (HuRIC)[14] describes a list of spoken commands between humans and robots, annotated using semantic frames and holistic spatial semantics, which has been explored by grammar-based and machine learning techniques that take advantage of both formalisms in which the data is described [9], [15].

Frequently robot-oriented tasks require special attention to the descriptions of spatial features and movements, which are at the centre of the contribution of Artzi et al. [6] and Tellex et al. [3]. Moreover, in 2014, SemEval hosted a task to parse commands targeting the control of a robot arm that moves objects around a board [7]. The work of Thomason et al. [16] also focuses on spatial representation aspects, covering, however, the acquisition of new vocabulary from dialogue with humans. In contrast to addressing a vocabulary gap, the challenge of these tasks relies on tracking expressions of direction and movement.

In the context of natural language programming, Azaria et al. [4] present a work on the task of mobile programming, dealing with five commands and focusing its evaluation on user experience aspects. Another initiative is the work of Neelakantan et al [5], which purposes a machine learning model to interpret questions against spreadsheets tasks which require the application of functional operations. Their contribution focus on the inference of a compositionality model for the operations, and the evaluation setup is constrained to a template-based syntax and a closed vocabulary set.

More recently Sales et al. [8] proposed a data set to deal with natural language programming which differs from others work in at least two aspects. First, the test collection presents a high variability in both vocabulary and grammar structure, when compared to the others test collection available so far. Secondly, the test collection is composed of small training sets for each frame, require the application of semantic parsing methods which can operate over small annotated data sets [17].

In this paper, we use this test collection to motivate and evaluate the semantic parsing approach. The test collection contains sets of natural language commands and the associated action frames corresponding to Web APIs, with the corresponding mappings. The goal of this work is to develop a model to map natural language command to action frames under those restrictions.

III. MAPPING NATURAL LANGUAGE COMMANDS TO ACTION FRAMES

The semantic parsing of natural language commands consists of mapping a natural language command to a formal function representation from a knowledge base. This function representation, named in the context of this work as action frame, is defined as a n-ary predicate-argument structure which describes a function interface (or signature) within a software system. In addition to the identification of which action frames the command refers to, the mapping process also identifies its parameters and their values (if any).

Taking as an example the natural language command which verbalises the user’s intent:

Write to newton@sebastian.com asking him to take a look at the NYT today.

In the example, the natural language command targets the specific action frame named send an email present in the knowledge base. Besides identifying the intended action frame, the semantic parser also needs to isolate newton@sebastian.com and take a look at the NYT today as argument values and recognise to which parameters (respectively, to address and message in this case) they should be assigned (among those offered by the action frame).

In the context of this paper, we name action instance the instantiation of an action call which describes the action frame itself along with values for its parameters, mapped partially or totally, as exemplified below:

Action instance: send an email
provider: Gmail
params:
message: “take a look at the NYT today”
to address: “newton@sebastian.com”

We formalise the target problem as follows. Let $A$ be a knowledge base (KB) composed of a set of $k$ action frames $(a_1, a_2, \ldots, a_k)$. Let $a_i = (n_i, l_i, P_i)$ be an element of $A$, where $n_i$ is the action’s name, $l_i$ is the action’s provider (a major object, service or functionality associated with the action), and $P_i$ is the set of the action’s parameters. Let $a_i^j$ be an instance of $a_i$, which also holds values for their parameters,
The Seq2Seq Setting receives as input a set of features related to the action description and the set of command objects, whereas the output defines the action frame together with their parameters values. The Classification Setting on its turn defines a model that receives as input features related to action instances and produces a class that defines the level of representativeness of the user intent.

totally or partially. Let $c_j$ be a natural language command which semantically represents a target action instance $a'_j$. The goal is to build a model which, given a set of action frames $A$ and a natural language command $c$, returns a list $B$ of ordered action instances.

Considering its nature, we can interpret this problem as a translation of the natural language command to an action instance. A typical method for solving this task is the sequence-to-sequence (Seq2Seq) machine learning model [18] which has represented the state-of-the-art for several natural language processing tasks [19], [20], [21]. The seq2seq model is designed to provide the target action frame and their set of parameters values simultaneously. As further detailed in Section V, this model did not succeed to solve the target problem, thereby our work concentrated on finding an alternative architecture to do so.

IV. PROPOSED APPROACH

The proposed approach is based on the intuition that the lower the training data is, the simpler the learning task should be[11]. Hence, instead of inducing the action instance directly, predicting at once both the action frame and its parameters values, we generate possible action instances and let the new model responsible for classifying them as representative or not of the user intent.

Figure 1 compares the input and output of the seq2seq architecture to the proposed remodelled architecture. The degree of relevance provided by the classifier together with other semantic features serves as input for a ranking model which places the relevant action instances according to its likelihood to represent the user intent.

For example, given the natural language command *Exchange 1000 Chilean Pesos to Euro*, the model previously generate a set of candidate action instances, as depicted in Table I, and then classifies them according to its relevance. This output is consequently used in conjunction with a ranking model as detailed in the next sections.

The semantic parsing approach relies on the use of frame embeddings where actions and parameters are embedded in distributional vector spaces. Geometric operators within the vector spaces such as distance and density are used as an input to the relevance ranking.

This approach got inspired by the work of Freitas [22] and Sales et al. [23], whereas semantic similarity metrics are used to match relevant content.

A. Semantic Parsing Steps

The semantic parsing is composed of four steps:

1) **Semantic Role Labelling:** In the first step, represented by Equation 1, the model reduces the natural language command $(c)$ to a lightweight representation composed of an action descriptor $(d)$ and a set of command objects $(O)$ using a shallow parser;

2) **Pivoting:** Next, the pivoting function, represented by Equation 2, aims at selecting a set of actions $\hat{A}$, where $\hat{A} \subset A$, $|\hat{A}| << |A|$ and $(\forall \hat{a} \in \hat{A}, \hat{a} \approx c)$, in the sense that the cardinality of the selected sub set is significantly smaller than the original one, and those elements are semantically related to the natural language command. As the data set is small, this step reduces the search space, maximising the probability of matching the parameters;

3) **Action Candidate Instance Generation:** Further, the third step generates the action instance candidates and represents them according to the feature set described in Section IV-A3;

4) **Relevance Ranking:** The last step classifies the candidate action instances and ranks them to the final user.

The four steps are formalised in the following equations:

\[
\sigma(c) = (d, O) \quad (1)
\]

\[
\rho(d, O, A) = \hat{A} \quad (2)
\]

\[
features(d, O, A) = Z \quad (3)
\]

\[
y = classify(Z) \quad (4)
\]

In the following sections we describe the shallow parser $\sigma(c)$, the pivoting function $\rho(d, O, A)$, the extraction of features and the final semantic parsing model.
<table>
<thead>
<tr>
<th>natural language command (q)</th>
<th>action descriptor (d)</th>
<th>set of command objects ( {o_1, o_2, \ldots, o_k} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exchange 1000 Chilean Pesos to Euro</td>
<td>Exchange</td>
<td>( (1000, \text{Chilean Pesos}, \text{Euro}) )</td>
</tr>
<tr>
<td>Send questions.doc to <a href="mailto:sandra@andrade.com">sandra@andrade.com</a></td>
<td>Send</td>
<td>( \text{/questions.doc, <a href="mailto:sandra@andrade.com">sandra@andrade.com</a>} )</td>
</tr>
<tr>
<td>Find an image of the Sputnik-1 on Flickr</td>
<td>Find image</td>
<td>( \text{(image, Sputnik-1, Flickr)} )</td>
</tr>
<tr>
<td>Translate file.txt from German to English.</td>
<td>Translate</td>
<td>( \text{(file.txt, German, English)} )</td>
</tr>
</tbody>
</table>

TABLE II: Examples of natural language commands and their respective action descriptor and set of command objects which are produced as result of the \( \sigma \) function.

1) Shallow Parsing: Equation 1 formally represents the \( \sigma \) function, where \( c \) is a natural language command, \( d \) is the action descriptor and \( O \) is the set of command objects.

The action descriptor is the minimal subset of tokens present in the natural language command which plays a key role in identifying the target action frame, usually corresponding to the main verb. The set of command objects comprises potential descriptors or values for the parameters, also including indirect speeches. Table II depicts examples of natural language commands and their representation as action descriptors and sets of command objects.

We implemented a simple but effective shallow parsing based on an explicit grammar defined on the top of the dependency tree and part-of-speech tags of the natural language command. The shallow parser assumes the first verbal phrase as the action descriptor and identifies the command objects according to the rules listed in Table III. In addition to the grammar, we classify indirect speeches as command objects using the algorithm proposed in [24].

For each command objects, we associate a semantic type, which is assigned by a named entity recogniser, whose implementation combines POS-tag rules with a gazetteer. It works by searching for the longest chain of tokens that maps to an element of the gazetteer, ignoring the tokens that are part of indirect speech. For the experiments, we developed a gazetteer using a subset of the DBpedia entities.

<table>
<thead>
<tr>
<th>Condition ( \phi(E) = )</th>
<th>Resulted CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi(\text{poly}) ) ( \phi(\text{dobj}) ) ( \phi(\text{nsubj}) )</td>
<td>( E_{\text{origin}} )</td>
</tr>
<tr>
<td>( \phi(\text{poss}) ) ( \phi(\text{amod}) )</td>
<td>( E_{\text{origin}} + E_{\text{dest}} )</td>
</tr>
</tbody>
</table>

TABLE III: The dependency-tree based rules to identify command objects (CO). Be a dependency tree a tuple \( H = (V, E, \phi) \), where: \( V \) is a finite set of nodes each of them representing tokens; \( E \subseteq V \times V \) is a finite set of edges, where \( E_{\text{origin}} \) represents the node in the origin and \( E_{\text{dest}} \) the node in the destination; and \( \phi : E \rightarrow C \) assigns a label from \( C \) to each edge.

2) Pivoting: The goal of the pivoting function \( \rho \) (Equation 2) is to establish the set of relevant action frames.

The function operates in a hyperspace defined by a distributional semantic model in which both the lightweight command representation and the action KB are projected as depicted in Figure 2. As the distributional semantic model provides vector representation only for single words, the representation of the command and actions are generated by composing the vectors of the words present in their descriptions.

The pivoting function defines the set of relevant action frames (\( \hat{\mathcal{A}} \)) by calculating a geometric measure in the hyperspace as shown in Figure 2. Then, each action frame into the area leads to another space where the command objects and the action parameters are projected based on the semantic representation of their names, types and densities. Considering the relation many-to-many between action parameters \( i \) (the parameters present in the action) and command objects \( j \) (the candidates values in the command), each pair [action frame, command] generates a set of action instances resulted from the permutation \( \mathcal{P}_j \).

3) Feature Extraction: The classification model interprets the geometric measure as an indicator of semantic relatedness. Consequently, the model represents each action instance by the
following list of features:

- $\cos(\vec{d}, \vec{n})$: The semantic relatedness between the action description ($d$) and the action name ($n$);
- $\max_{0 \leq i \leq m} \cos(\vec{o}^{\text{literal}}_i, \vec{n})$: The maximum semantic relatedness between the command objects ($o$) and the provider ($n$);
- $\cos(\vec{o}^{\text{literal}}_i, \vec{n})$: The set of semantic relatedness between the pairs composed of the command objects ($o$) and action parameter ($n$);
- $\cos(\vec{o}^{\text{type}}_i, \vec{n})$: The set of semantic relatedness between the pairs composed of the semantic type of the parameter description ($o$) and the action parameter ($n$);
- $\text{den}(p_i)$: The set of the densities of the action parameters.

These semantic relatedness scores are used as input features to identify jointly the most relevant action frame and the best configuration of parameters values.

4) Matching Model: The matching model classifies the action frame instances into:

- (i) wrong frame (score 0);
- (ii) right frame with wrong parameters (score 1);
- (iii) right frame with partial right parameters (score 2);
- (iv) right frame with right parameters (score 3).

This small, but discriminative set of classes works as a data augmentation method, in the sense that it enables the existence of many training instances of the same class, even considering the small training data set the task offers.

Equation 5 defines the score function from which the ranking is generated, where the classify function receives as input the set of features disposed as a unique vector $z$.

$$\text{classify}(z) * 1000 + \sum_{i=0}^{n}(z_i)$$  \hspace{1cm} (5)

The multiplication in the score function guarantees that action instances from a higher class is always ranked above those classified in lower classes.

V. Experiments

Besides the mapping of natural language commands to action frames, the original Task 11 at SemEval 2017, demanded the construction of a logic algorithm and the resolution of parameters’ co-references[8]. It is not a coincidence that only one team participated in the challenge and this team had succeeded in less than 6% of the test collection[17]3. Despite the relevance of the proposed challenge, its level of granularity makes the problem too ambitious given the volume of data released.

To focus on the mapping of natural language commands to action frames, two curators reformulated the original task by (i) removing co-references and (ii) isolating and merging the original commands in such a way that privileged variance in vocabulary and syntax structure.

<table>
<thead>
<tr>
<th>Action name</th>
<th>Provider</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a status message</td>
<td>Facebook</td>
<td>status message</td>
</tr>
<tr>
<td>Currency converter</td>
<td>null</td>
<td>from amount, from, to</td>
</tr>
<tr>
<td>Open garage door</td>
<td>Garageio</td>
<td>Which door</td>
</tr>
<tr>
<td>Create an issue</td>
<td>GitHub</td>
<td>repository, title, body</td>
</tr>
<tr>
<td>Create new contact</td>
<td>Google Contacts</td>
<td>full name, email...</td>
</tr>
</tbody>
</table>

TABLE IV: Examples of action frames present in the KB.

A. Data

After this process, the test collection ended up with 185 natural language commands. With regard to the action knowledge base, we kept those extracted from IFTTT and others referenced in the evaluation scenarios, adapting some of them to guarantee accurate semantic description, which resulted in a KB of 2005 action frames. The new data set composed of the action KB and the mappings of natural language commands to action frames is available at https://rebrand.ly/nlc-dataset.

Table IV shows some examples of action frames present in the action KB.

Given the similarity to our target problem, we considered the use of the data set defined by Quirk et al. [25] which was also extracted from the ifttt.com platform. The data set comprises descriptions of if-then recipes provided by users. However, the task requires only the identification of the action frames, keeping aside the instantiation of the parameter values. This limitation cannot be overcome since the data set does not contain the parameters values, nor there is any guarantee that those values are explicitly present in the recipe descriptions.

As a baseline, we designed the experiments to evaluate both a Seq2Seq model and the proposed semantic parsing method.

To fix the problem of imbalanced classes in the training data, we applied the random majority under-sampling with replacement, making use of the imbalanced-learn library [26].

B. Sequence-to-Sequence Model

We implemented a Seq2Seq neural network composed of LSTM cells as recurrent unit in the style of [18]. The input layer was designed to receive a matrix $Z \in \mathbb{R}^n$ containing the set of instances, represented by the action’s descriptor and the set of command objects expressed as 300-length vectors extracted from the Google-News-300 Word2Vec model.

The training process was conducted in two steps. In the first step, we encoded each action frame in the KB as input, in order to make the model aware of every action frame. Secondly, the training data was applied in a 10-fold cross-validation fashion.

We evaluated different network architectures varying the number of layers (1 to 3), nodes (1 to 3 times the input size), epochs (up to 500), learning rates (0.001, 0.003, 0.01, 0.03, 0.1) and batch sizes (100%, 50% and 25% of the training data).

All the evaluated Seq2Seq models delivered a F1-Score of 0.

3Considering the mapping of action frames and parameter values.
C. The Proposed Model

To evaluate the proposed model, we instantiated different implementations for both the pivoting function and the classifier. The pivoting function assumed three implementations:

- **Identity**: In the first case, ρ implements the identity function. This configuration means that no filter is applied and aims at measure the relevance of the pivoting step;

- **TF/IDF**: A natural candidate for a pivoting function is the TDF/IDF weighting scheme, which conditions the target actions to those that overlaps vocabulary with the query. TDF/IDF pivoting function in average limits the number of target action frames to 10;

- **Nearest Neighbours**: The third approach uses an nearest neighbours method to select the 50 closest action frames, when projecting the natural language command into the distributional semantic space. This type of function is not limited to the overlapping of vocabulary, but expands their relation to the latent notion of semantics defined by the distributional vector model.

With regard to the classifier, we evaluated three learning methods: Random Forest, Support Vector Machine and a simple Multilayer Perceptron Neural Network (MLP). We evaluated each learning method in different fashions, identifying their hyperparameters by grid search. For Random Forest, the number of estimators ranged into (100, 300, 1000, 3000, 10000), the maximum number of features assumed the sqrt or the log2 of the total available. For the Support Vector Machine classifier the grid search was applied considering the kernel varying into linear, sigmoid and polynomial (with 2, 3, 5 degrees) and gamma into the logspace(-9, 3, 3), keeping a fixed C=1. Finally, for the MLP network, we evaluated under the same variation specified for the Seq2Seq model. The classifiers receive the input as described in Section IV-A3.

We evaluate our architecture purposely with off-the-shelf implementations for both the pivoting function and the classifier. This decision seeks to highlight the relevance of the feature selection and model architecture to the final solution.

Our experiments used the skip-gram model generated over the Google news data set as the distributional space model [27]. To speed up the development, we used the Indra word embedding server [28].

VI. RESULTS AND DISCUSSION

Table V shows the results of the proposed approach in different combinations of pivoting functions and classifiers, measured in relation to the recall and mean reciprocal rank (MRR). The evaluation was carried out in two scenarios: the first considers the actions ranked up to the 10th position, whereas the second, up to the 50th. In the experiments, we assumed that only one action frame corresponded to the target answer. This assumption makes precision a redundant indicator since it can be derived from the recall.

The application of the identity function as a pivoting function represents the absence of a pivoting area in the sense that the full action KB is considered in the classification step. Given the simplicity of the feature set and the straightforwardness of the matching model, the results show that the pivoting function plays a key role in increasing the accuracy. The experiments in which the identity function is present consistently deliver lower recall, as shown in Figure 3. Even when combined with Random Forest, where the identity function had a more competitive performance in recall, Figure 4a shows the target action instances are significantly lower ranked.

The other two pivoting functions have similar results in recall, with a slight advantage to Nearest Neighbours from TF/IDF. This is explained by the fact that the TF/IDF ignores relevant action frames more often than Nearest Neighbours.

Random Forest is by far the best classifier considering either recall or MRR in all of the evaluation scenarios, whereas the SVM and MLP classifiers perform similarly in relation to the recall. Our assumptions is that the set of semantic relatedness features shows a high level of independence, producing many sub-optimal areas that can be better avoided by a tree-based learning model. This assumption is reinforced considering the scenarios in conjunction with the identity function. The higher volume of data tends to generate more sub-optimal spaces and represents exactly the scenario where the classifiers shows larger gaps.

We conjecture that the Seq2Seq approach overloads the learning model demanding the identification of the target action over a set of thousands of frames, besides the correct map of the parameters values. The low relation between the number of classes vs. the number of training examples, which explain its failure.

In conclusion, the combination of a random-forest classifier with the nearest-neighbours pivoting function was able to solve up to 68% considering the TOP-10 and up to 85% when considering the TOP-50, with the MRR scoring around 0.3, which means that the target action instances are placed in

<table>
<thead>
<tr>
<th>Classifier/Pivoting function</th>
<th>Scenario</th>
<th>Identity recall</th>
<th>Identity MRR</th>
<th>TF/IDF recall</th>
<th>TF/IDF MRR</th>
<th>Nearest Neighbours recall</th>
<th>Nearest Neighbours MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>TOP-10</td>
<td>0.4217</td>
<td>0.1264</td>
<td>0.6594</td>
<td>0.2878</td>
<td>0.6825</td>
<td>0.3038</td>
</tr>
<tr>
<td></td>
<td>TOP-50</td>
<td>0.2560</td>
<td>0.0123</td>
<td>0.1877</td>
<td>0.0774</td>
<td>0.3557</td>
<td>0.5120</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>TOP-10</td>
<td>0.0476</td>
<td>0.0099</td>
<td>0.4298</td>
<td>0.1901</td>
<td>0.3608</td>
<td>0.1207</td>
</tr>
<tr>
<td></td>
<td>TOP-50</td>
<td>0.2280</td>
<td>0.0123</td>
<td>0.6224</td>
<td>0.1978</td>
<td>0.6232</td>
<td>0.1187</td>
</tr>
<tr>
<td>MLP - Neural Network</td>
<td>TOP-10</td>
<td>0.0738</td>
<td>0.0187</td>
<td>0.4798</td>
<td>0.2063</td>
<td>0.3944</td>
<td>0.1323</td>
</tr>
<tr>
<td></td>
<td>TOP-50</td>
<td>0.3110</td>
<td>0.0311</td>
<td>0.7015</td>
<td>0.2171</td>
<td>0.7292</td>
<td>0.1514</td>
</tr>
</tbody>
</table>

TABLE V: Recall and mean reciprocal rank for the combination of different pivoting functions and classifiers evaluated in the TOP-10 and TOP-50 scenarios.
average at the 3rd or 4th positions.

VII. Conclusion

In this paper we propose a semantic parsing method to map natural language commands to action frames for large and heterogeneous frame sets under a restricted set of annotated data. The proposed distributional semantic parsing method operating with a nearest-neighbours pivoting and a random forest alignment quality classifier achieved a recall of 0.682 and a mean reciprocal rank of 0.303 for the TOP-10 results over a knowledge base of 2005 distinct action frames.

As a future work, we plan to analyse the suitability of the proposed semantic parsing model as a foundation for the definition of a search-and-run programming model.

Acknowledgment

This publication has emanated from research partially supported by the National Council for Scientific and Technological Development, Brazil (CNPq).

References


