A Mapping-Based Approach for General Formal Human Computer Interaction Using Natural Language

Vincent Letard
LIMSI CNRS
letard@limsi.fr

Sophie Rosset
LIMSI CNRS
rosset@limsi.fr

Gabriel Illouz
LIMSI CNRS
illouz@limsi.fr

Abstract

We consider the problem of mapping natural language written utterances expressing operational instructions[1] to formal language expressions, applied to French and the R programming language. Developing a learning operational assistant requires the means to train and evaluate it, that is, a baseline system able to interact with the user. After presenting the guidelines of our work, we propose a model to represent the problem and discuss the fit of direct mapping methods to our task. Finally, we show that, while not resulting in excellent scores, a simple approach seems to be sufficient to provide a baseline for an interactive learning system.

1 Introduction

Technical and theoretical advances allow achieving more and more powerful and efficient operations with the help of computers. However, this does not necessarily make it easier to work with the machine. Recent supervised learning work (Allen et al., 2007; Volkova et al., 2013) exploited the richness of human-computer interaction for improving the efficiency of a human performed task with the help of the computer.

Contrary to most of what was proposed so far, our long term goal is to build an assistant system learning from interaction to construct a correct formal language (FL) command for a given natural language (NL) utterance, see Table 1. However, designing such a system requires data collection, and early attempts highlighted the importance of usability for the learning process: a system that is hard to use (*eg.* having very poor performance) would prevent from extracting useful learning examples from the interaction. We thus need to provide the system with a basis of abilities and knowledge to allow both incremental design and to keep the interest of the users, without which data turn to be way more tedious to collect. We assume that making the system usable requires the ability to provide help to the user more often than it needs help from him/her, that is an accuracy over 50%.

We hypothesize that a parametrized direct mapping between the NL utterances and the FL commands can reach that score. A knowledge set K is built from parametrized versions of the associations shown in Table 1. The NL utterance U_{best} from K that is the closest to the request-utterance according to a similarity measure is chosen and its associated command $C(U_{best})$ is adapted to the parameters of the request-utterance and returned. For example, given the request-utterance U_{req}: *"Load the file data.csv"*, the system should rank the utterances of K by similarity with U_{req}. Considering the associations represented in Table 1, the first utterance should be the best ranked, and the system should return the command:
"`var1 <- read.csv("data.csv")`".
Note that several commands can be proposed at the same time to give the user alternate choices.

We use Jaccard, tf-idf, and BLEU similarity measures, and consider different selection strategies. We highlight that the examined similarity measures show enough complementarity to permit the use of combination methods, like vote or statistical classification, to improve *a posteriori* the efficiency of the retrieval.

2 Related Work

2.1 Mapping Natural Language to Formal Language

Related problems have been previously processed using different learning methods. Branavan (2009,

[1]We call *operational instruction* the natural language expression of a command in any programming language.

	NL utterances	FL commands (in R)
1	Charge les données depuis **”res.csv”** Load the data from **”res.csv”**	`var1=read.csv("res.csv")`
2	Trace l’histogramme de la colonne **2** de **tab** Draw a bar chart with column **2** of **tab**	`plot(hist(tab[[2]]))`
3	Dessine la répartition de la colonne **3** de **tab** Draw the distribution of column **3** of **tab**	`plot(hist(tab[[3]]))`
4	Somme les colonnes **3** et **4** de **tab** Compute the sum of columns **3** and **4** of **tab**	`var2=c(sum(tab[3]),sum(tab[4]))`
5	Somme les colonnes **3** et **4** de **tab** Compute the sum of columns **3** and **4** of **tab**	`var3=sum(c(tab[[3]],tab[[4]]))`

Table 1: A sample of NL utterances to FL commands mapping
These examples specify the expected command to be returned for each utterance. The tokens in bold font are linked with the commands parameters, cf. section 4. Note that the relation between utterances and commands is a n to n. Several utterances can be associated to the same command and conversely.

2010) uses reinforcement learning to map English NL instructions to a sequence of FL commands. The mapping takes high-level instructions and their constitution into account. The scope of usable commands is yet limited to graphical interaction possibilities. As a result, the learning does not produce highly abstract schemes. In the problematic of interactive continuous learning, Artzi and Zettlemoyer (2011) build by learning a semantic NL parser based on combinatory categorial grammars (CCG). Kushman and Barzilay (2013) also use CCG in order to generate regular expressions corresponding to their NL descriptions. This constructive approach by translation allows to generalize over learning examples, while the expressive power of regular expressions correspond to the type-3 grammars of the Chomsky hierarchy. This is not the case for the programming languages since they are at least of type-2. Yu and Siskind (2013) use hidden Markov models to learn a mapping between object tracks from a video sequence and predicates extracted from a NL description. The goal of their approach is different from ours but the underlying problem of finding a map between objects can be compared. The matched objects constitute here a FL expression instead of a video sequence track.

2.2 Machine Translation

Machine translation usually refers to transforming a NL sentence from a source language to another sentence of the same significance in another natural language, called target language. This task is achieved by building an intermediary representation of the sentence structure at a given level of abstraction, and then encoding the obtained object into the target language. While following a different goal, one of the tasks of the XLike project (Marko Tadić et al., 2012) was to examine the possibility of translating statements from NL (English) to FL (Cycl). Adapting such an approach to operational formal target language can be interesting to investigate, but we will not focus on that track for our early goal.

2.3 Information Retrieval

The issue of information retrieval systems can be compared with the operational assistant’s (OA), when browsing its knowledge. Question answering systems in particular (Hirschman and Gaizauskas, 2001), turn out to be similar to OA since both types of systems have to respond to a NL utterance of the user by generating an accurate reaction (which is respectively a NL utterance containing the wanted information, or the execution of a piece of FL code). However, as in (Toney et al., 2008), questions answering systems usually rely on text mining to retrieve the right information. Such a method demands large sets of annotated textual data (either by hand or using an automatic annotator). Yet, tutorials, courses or manuals which could be used in order to look for responses for operational assistant systems are heterogeneous and include complex or implicit references to operational knowledge. This makes the annotation of such data difficult. Text mining methods are thus not yet applicable to operational assistant systems but could be considered once some annotated data is collected.

3 Problem Formulation

As we introduced in the first section, we represent the knowledge K as a set of examples of a binary relation $R : NL \rightarrow FL$ associating a NL utterance to a FL command. If we consider the simple case of a functional and injective relation, each utterance is associated to exactly one command. This is not realistic since it is possible to reformulate nearly any NL sentence. The case of a non injective relation covers better the usual cases: each command can be associated with one or more utterances, this situation is illustrated by the second and third examples of Table 1. Yet, the real-life case should be a non injective nor functional relation. Not only multiple utterances can refer to a same command, but one single utterance can also stand for several distinct commands (see the fourth and fifth examples[2] in Table 1). We must consider all these associations when matching a request-utterance U_{req} for command retrieval in K.

At this point, several strategies can be used to determine what to return, with the help of the similarity measure $\sigma : NL \times NL \rightarrow \mathbb{R}$ between two NL utterances. Basically, we must determine if a response should be given, and if so how many commands to return. To do this, two potential strategies can be considered for selecting the associated utterances in K.

The first choice focuses on the number of responses that are given for each request-utterance. The n first commands according to the rankings of their associated utterances in K are returned. The rank r of a given utterance U is computed with:

$$r(U|U_{req}) = \left|\left\{U' \in K : \sigma(U_{req}, U') > \sigma(U_{req}, U)\right\}\right| \tag{1}$$

The second strategy choice can be done by determining an absolute similarity threshold below which the candidate utterances from K and their associated sets of commands are considered too different to match. The resulting set of commands is given by:

$$Res = \{C \in FL : (U,C) \in K, \sigma(U_{req}, U) < t\} \tag{2}$$

with t the selected threshold. Once selected the set of commands to be given as response, if there are more than one, the choice of the one to execute can be done interactively with the help of the user.

[2]The command 4 returns a vector of the sums of each column, while the command 5 returns the sum of the columns as a single integer.

4 Approach

We are given a simple parsing result of both the utterance and the command. The first step to address is the acquisition of examples and the way to update the knowledge. Then we examine the methods for retrieving a command from the knowledge and a given request-utterance.

Correctly mapping utterances to commands requires at least to take their respective parameters into account (variable names, numeric values, and quoted strings). We build generic representations of utterances and commands by identifying the parameters in the knowledge example pair (see Table 1), and use them to reconstruct the command with the parameters of the request-utterance.

4.1 Retrieving the Commands

We applied three textual similarity measures to our model in order to compare their strengths and weaknesses on our task: the Jaccard similarity coefficient (Jaccard index), a tf-idf (Term frequency-inverse document frequency) aggregation, and the BLEU (Bilingual Evaluation Understudy) measure.

4.1.1 Jaccard index

The Jaccard index measures a similarity between two sets valued in the same superset. For the present case, we compare the set of words of the input NL instruction and the one of the compared candidate instruction, valued in the set of possible tokens. The adapted formula for two sentences S_1 and S_2 results in:

$$J(s_1, s_2) = \frac{|W(s_1) \cap W(s_2)|}{|W(s_1) \cup W(s_2)|} \tag{3}$$

where $W(S)$ stands for the set of words of the sentence S. The Jaccard index is a baseline to compare co-occurences of unigrams, and should be efficient mainly with corpora containing few ambiguous examples.

4.1.2 tf-idf

The tf-idf measure permits, given a word, to classify documents on its importance in each one, regarding its importance in the whole set. This measure should be helpful to avoid noise bias when it comes from frequent terms in the corpus. Here, the documents are the NL utterances from K, and they are classified regarding the whole request-utterance, or input sentence s_i. We then use the following aggregation of the tf-idf values for each word of $\overline{s_i}$.

$$tfidf_S(s_i, s_c) = \frac{1}{|W(s_i)|} \sum_{w \in W(s_i)} tfidf(w, s_c, S) \quad (4)$$

with $S = \{s | (s, com) \in K\}$, where s_i is the input sentence, $s_c \in S$ is the compared sentence, and where the tf-idf is given by:

$$tfidf(w, s_c, S) = f(w, s_c) idf(w, S) \quad (5)$$

$$idf(w, S) = log \left(\frac{|S|}{|\{s \in S | w \in s\}|} \right) \quad (6)$$

where at last $f(w, s)$ is the frequency of the word w in the sentence s. As we did for the Jaccard index, we performed the measures on both raw and lemmatized words. On the other hand, getting rid of the function words and closed class words is not here mandatory since the tf-idf measure already takes the global word frequency into account.

4.1.3 The BLEU measure

The bilingual evaluation understudy algorithm (Papineni et al., 2002) focuses on n-grams co-occurrences. This algorithm can be used to discard examples where the words ordering is too far from the candidate. It computes a modified precision based on the ratio of the co-occurring n-grams within candidate and reference sentences, on the total size of the candidate normalized by n.

$$P_{BLEU}(s_i, S) = \sum_{gr_n \in s_i} \frac{\max_{s_c \in S} occ(gr_n, s_c)}{grams(s_i, n)} \quad (7)$$

where $grams(s, n) = |s| - (n - 1)$ is the number of n-grams in the sentence s and $occ(gr_n, s) = \sum_{gr_n{}' \in s} [gr_n = gr_n{}']$ is the number of occurrences of the n-gram gr_n in s. BLEU also uses a brevity penalty to prevent long sentences from being too disadvantaged by the n-gram based precision formula. Yet, the scale of the length of the instructions in our corpus is sufficiently reduced not to require its use.

4.2 Optimizing the similarity measure

We applied several combinations of filters to the utterances compared before evaluating their similarity. We can change the set of words taken into account, discarding or not the non open-class words[3]. Identified non-lexical references such as variable names, quoted character strings and numeric values can also be discarded or transformed to standard substitutes. Finally, we can apply or not a lemmatization[4] on lexical tokens.By discarding non open-class words, keeping non-lexical references and applying the lemmatization, the second utterance of Table 1 would then become:

draw bar chart column xxVALxx xxVARxx

5 Experimental Setup

5.1 Parsing

The NL utterances first pass through an arithmetic expression finder to completely tag them before the NL analyzer. They are then parsed using WMATCH, a generic rule-based engine for language analysis developed by Olivier Galibert (2009). This system is modular and dispose of rules sets for both French and English. As an example, the simplified parsing result of the first utterance of Table 1 looks like:

```
<_operation>
  <_action> charge|_˜V </_action>
  <_det> les </_det>
  <_subs> données|_˜N </_subs>
  <_prep> depuis </_prep>
  <_unk> "res.csv" </_unk>
</_operation>
```

Words tagged as unknown are considered as potential variable or function names. We also added a preliminary rule to identify character strings and count them among the possibly linked features of the utterance. The commands are normalized by inserting spaces between every non semantically linked character pair and we identify numeric values, variable/function names and character strings as features.

Only generative forms of the commands are associated to utterances in the knowledge. This form consists in a normalized command with unresolved references for every parameter linked with the learning utterance. These references are resolved at the retrieving phase by matching with the tokens of the request-utterance.

5.2 Corpus Constitution

Our initial corpus consists in 605 associations between 553 unique NL utterances in French and 240 unique R commands.

[3]Open-class words include nouns, verbs, adjectives, adverbs and interjections.

[4]Lemmatization is the process of transforming a word to its canonical form, or lemma, ignoring the inflections. It can be performed with a set of rules or with a dictionary. The developed system uses a dictionary.

The low number of documents describing a majority of R commands and their heterogeneity make automatic example gathering not yet achievable. These documentations are written for human readers having global references on the task. Thus, we added each example pair manually, making sure that the element render all the example information and that the format correspond to the corpus specifications. Those specifications are meant to be the least restrictive, that is: a NL utterance must be written as to ask for the execution of the associated R task. It therefore should be mostly in the imperative form and reflect, for experienced people, a usual way they would express the concerned operation for non specialists.

5.3 Evaluation Metrics

The measures that can contribute to a relevant evaluation of the system depend on its purpose. Precision and recall values of information retrieval systems are computed as follows:

$$P = \frac{\text{\# correct responses}}{\text{\# responses given}} \tag{8}$$

$$R = \frac{\text{\# correct responses}}{\text{\# responses in K}} \tag{9}$$

Note that the recall value is not as important as for information retrieval: assuming that the situation showed by the fourth and fifth associations of Table 1 are not usual[5], there should be few different valid commands for a given request-utterance, and most of them should be equivalent. Moreover, the number of responses given is fixed (so is the number of responses in K), the recall thus gives the same information as the precision, with a linear coefficient variation.

These formulae can be applied to the "command level", that is measuring the accuracy of the system in terms of its good command ratio. However, the user satisfaction can be better measured at the "utterance level" since it represents the finest granularity for the user experience. We define the utterance precision uP as:

$$uP = \frac{\text{\# correct utterances}}{\text{\# responses given}} \tag{10}$$

where "# correct utterances" stands for the number of request-utterances for which the system provided at least one good command.

[5]Increasing the tasks covering of the corpus will make these collisions more frequent, but this hypothesis seems reasonable for a first approach.

6 Results and Discussion

The system was tested on 10% of the corpus (61 associations). The set of known associations K contains 85% of the corpus (514 associations), instead of 90% in order to allow several distinct drawings (40 were tested), and thus avoid too much noise.

6.1 Comparing similarity measures

As shown in Table 2 the tf-idf measure outperforms the Jaccard and BLEU measures, whichever filter combination is applied. The form of the utterances in the corpus causes indeed the repetition of a small set of words across the associations. This can explain why the inverse document frequency is that better.

non-lexical	included		not included	
lemmatize	yes	no	yes	no
Jaccard	36.5	36.5	21.2	23.0
tf-idf	**48.0**	**51.9**	**36.5**	**40.4**
BLEU	30.8	32.7	26.9	30.8
chance	1.9			

Table 2: Scores of precision by utterance (uP), providing 3 responses for each request-utterance.

The lemmatization and the inclusion of non open-class words (not shown here) does not seem to have a clear influence on uP, whereas including the non-lexical tokens allows a real improvement. This behaviour must result from the low length average (7.5 words) of the utterances in the corpus.

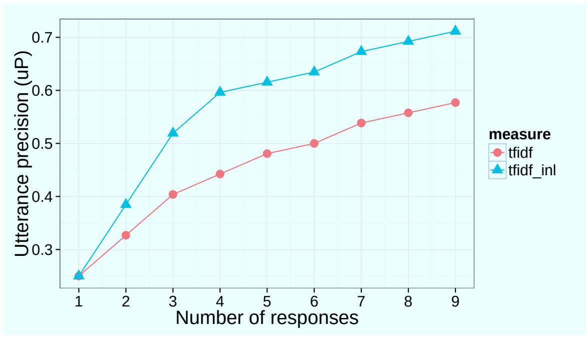

Figure 1: Utterance precision (uP) for a fixed number of responses by utterance. The tfidf_inl curve includes the non-lexical tokens.

Note that uP is obtained with Equation 10, which explains the increase of the precision along the number of responses.

Figure 1 shows the precision obtained with tfidf while increasing the number of commands given for each request-utterance. It comes out that it is useful to propose at least 3 commands to the user. It would not be interesting, though, to offer a choice of more than 5 items, because the gain on uP would be offset by the time penalty for retrieving the good command among the proposals.

6.2 Allowing silence

We also tested the strategy of fixing an absolute threshold to decide between response and silence. Given a request-utterance and an associated ordering of K according to σ, the system will remain silent if the similarity of the best example in K is below the defined threshold.

Surprisingly, it turned out that for every measure, the 6 best similar responses at least were all wrong. This result seems to be caused by the existence, in the test set of commands uncovered by K, of some very short utterances that contain only one or two lexical tokens.

6.3 Combinations

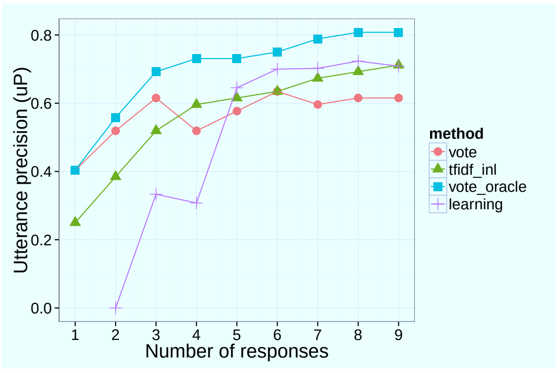

Figure 2: Comparison of the combinations with the tf-idf_inl method. Oracle and actual vote are done using tf-idf, Jaccard, and BLEU, with and without non-lexical tokens. The training set for learning is the result of a run on K.

Having tested several methods giving different results, combining these methods can be very interesting depending on their complementarity. The oracle vote using the best response among the 6 best methods shows an encouraging progression margin (*cf.* Figure 2). The actual vote itself outperforms the best method for giving up to 3 responses (reaching 50% for only 2 responses). However, the curve position is less clear for more responses, and tests must be performed on other drawings of K to measure the noise influence.

The complementarity of the methods can also be exploited by training a classification model to identify when a method is better than the others. We used the similarity values as features and the measure that gave a good response as the reference class label (best similarity if multiple, and "none" class if no good response). This setup was tested with the support vector machines using libsvm (Chang and Lin, 2011) and results are shown in Figure 2. As expected, machine learning performs poorly on our tiny corpus. The accuracy is under 20% and the system only learned when to use the best method, and when to give no response. Still, it manages to be competitive with the best method and should be tested again with more data and multiple drawings of K.

7 Conclusion and Future Work

The simple mapping methods based on similarity ranking showed up to 60% of utterance precision[6] remaining below a reasonable level of user sollicitation, which validate our prior hypothesis. A lot of approaches can enhance that score, such as adding or developing more suitable similarity measures (Achananuparp et al., 2008), combining learning and vote or learning to rerank utterances.

However, while usable as a baseline, these methods only allow poor generalization and really need more corpus to perform well. As we pointed out, the non-functionality of the mapping relation also introduces ambiguities that cannot be solved using the only knowledge of the system.

Thanks to this baseline method, we are now able to collect more data by developing an interactive agent that can be both an intelligent assistant and a crowdsourcing platform. We are currently developing a web interface for this purpose. Finally, situated human computer interaction will allow the real-time resolving of ambiguities met in the retrieval with the help of the user or with the use of contextual information from the dialogue.

Aknowledgements

The authors are grateful to every internal and external reviewer for their valuable advices. We also would like to thank Google for the financial support for the authors participation to the conference.

[6]The corpus will soon be made available.

References

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. *The Evaluation of Sentence Similarity Measures*. In Data Warehousing and Knowledge Discovery, Springer.

James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Tayson. 2007. *PLOW: A Collaborative Task Learning Agent*. In Proceedings of the 22nd National Conference on Artificial Intelligence.

Yoav Artzi, and Luke S. Zettlemoyer. 2011. *Bootstrapping semantic parsers from conversations*. Proceedings of the conference on empirical methods in natural language processing.

S.R.K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. 2010. *Reading Between the Lines: Learning to Map High-level Instructions to Commands*. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. *Reinforcement Learning for Mapping Instructions to Actions*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP.

Chih-Chung Chang, and Chih-Jen Lin. 2011. *LIBSVM: A Library for Support Vector Machines*. ACM Transactions on Intelligent Systems and Technology

Olivier Galibert. 2009. *Approches et méthodologies pour la réponse automatique à des questions adaptées à un cadre interactif en domaine ouvert*. Doctoral dissertation, Université Paris Sud XI.

Lynette Hirschman, and Robert Gaizauskas. 2001. *Natural language question answering: The view from here*. Natural Language Engineering 7. Cambridge University Press.

Nate Kushman, and Regina Barzilay. 2013. *Using Semantic Unification to Generate Regular Expressions from Natural Language*. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. *Bleu: a Method for Automatic Evaluation of Machine Translation*. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.

Marko Tadić, Božo Bekavac, Željko Agić, Matea Srebačić, Daša Berović, and Danijela Merkler. 2012. *Early machine translation based semantic annotation prototype* XLike project www.xlike.org .

Dave Toney, Sophie Rosset, Aurélien Max, Olivier Galibert, and éric Billinski. 2008. *An Evaluation of Spoken and Textual Interaction on the RITEL Interactive Question Answering System* In Proceedings of the Sixth International Conference on Language Resources and Evaluation.

Svitlana Volkova, Pallavi Choudhury, Chris Quirk, Bill Dolan, and Luke Zettlemoyer. 2013. *Lightly Supervised Learning of Procedural Dialog System* In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Haonan Yu, and Jeffrey Mark Siskind. 2013. *Grounded Language Learning from Video Described with Sentences*. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.