6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

TOWARD UNCONSTRAINED COMMAND AND CONTROL: DATA-DRIVEN SEMANTIC INFERENCE

Jerome R. Bellegarda and *Kim E.A. Silverman*

Spoken Language Group, Apple Computer, Inc.
Cupertino, California 95014, USA

ABSTRACT

Command and control tasks are typically approached using a context-free grammar as language model. While ensuring a good system performance, this imposes a rigid framework on users, by implicitly forcing them to conform to a pre-defined interaction structure. This paper introduces the concept of *data-driven semantic inference*, which in principle allows for any word constructs in command/query formulation. Each unconstrained word string is automatically mapped onto the intended action through a semantic classification against the set of supported actions. The underlying (latent semantic analysis) framework relies on co-occurrences between words and commands, as observed in a training corpus. A suitable extension can also handle commands that are ambiguous at the word level. Experiments conducted on a desktop command and control task involving 113 different actions show a classification error rate of 1.7%.

1. INTRODUCTION

Command and control tasks tend to be restricted to a well-defined domain of discourse, for two main reasons: (i) it increases the chance that acoustic data collection will maximize acoustic coverage for the application at hand, and (ii) it typically allows a context-free grammar (CFG) to be used as language model (LM), which tightly constrains the search for the most likely sequence of words uttered. The latter in turn has a dual benefit. First, it improves recognition performance by eliminating hypotheses which fall outside the grammar. Second, it simplifies the translation from recognized utterance to action to be taken, by establishing *a priori* an unambiguous mapping between parses and actions. As a result, many systems have reached commercial deployment, and routinely operate in speaker-independent mode under adverse environmental conditions (e.g., over the telephone, in conjunction with a far-field microphone, or in low signal-to-noise ratio situations).

From the service provider's point of view, the dual benefit just mentioned essentially justifies the cost of hand-crafting a new CFG for each application. From the user's point of view, the trade-off is not so clear. In the interest of efficiency, there is a natural tendency for the service provider to limit variability in the commands. But any utterance which is syntactically out-of-grammar will be either rejected or misrecognized, even though it might be semantically adequate for the intended action. As a result, the user is implicitly forced to conform to a pre-defined formulation structure, composed of an explicit and exclusive enumeration of canonical expressions for each supported action. This assumes a high level of user cooperation.

A substantial fraction of users may have difficulties and become frustrated with such a rigid behavior. For example, first-time users often do not know (or care about!) the exact syntax to use in formulating the commands. Forcing them to seek help contradicts the intuitive and natural nature of speech communication. Such users would likely adopt the technology more readily if they were able to speak to the system in their own words. This has led developers to construct ever bigger grammars, with many alternative paths to try to capture, for each command, as many anticipated variants as possible. This eventually has a negative impact on system accuracy and responsiveness, due to the increased size of the search space. But the biggest drawback is that this approach cannot generalize from the seen to the unseen. Because the next "naive" user could still come up with an unexpected variant, this route ultimately has only limited appeal.

A more flexible solution is to allow for an "unconstrained" formulation, much like in dictation applications, where the output is a string of transcribed words. In this situation, since a CFG is not (yet) a feasible alternative, a suitably trained n-gram LM must be used to constrain the search [1]. The resulting large vocabulary transcription will therefore contain word errors. Still, because of the limited domain of discourse, we can expect a reasonably low word error rate, which may not materially affect the basic meaning of the utterance. The main difficulty now lies in recognizing the action intended to be taken. In essence, this decision requires the extraction of sufficient semantic information from the transcription.

This paper describes a novel approach to this problem, referred to as (data-driven) *semantic inference*. In contrast with usual inference engines (cf. [2]), semantic inference does not rely on formal behavioral principles extracted from a knowledge base. Instead, the domain knowledge is automatically encapsulated in a data-driven fashion within a low-dimensional vector space. As a result, semantic inference can be thought of as a way to perform (albeit elementary) "bottom-up" natural language understanding. The paper is organized as follows. The next section contrasts semantic inference against conventional command and control. Section 3 and 4 describe the approach, based on the latent semantic analysis (LSA) framework [3]. In Section 5, we briefly mention a necessary extension to handle commands that are ambiguous at the word level. Finally, Section 6 reports experimental results illustrating the benefits of semantic inference.

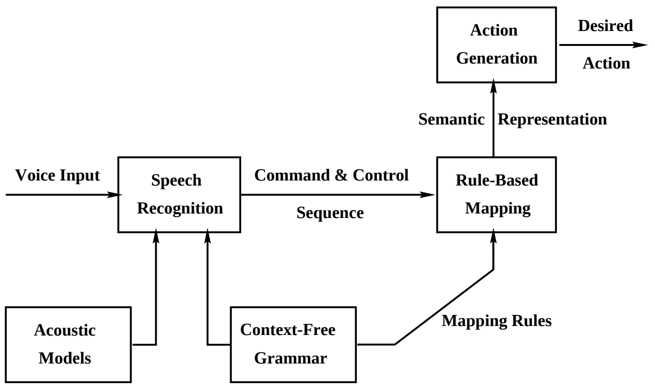

Fig. 1. Conventional Command and Control.

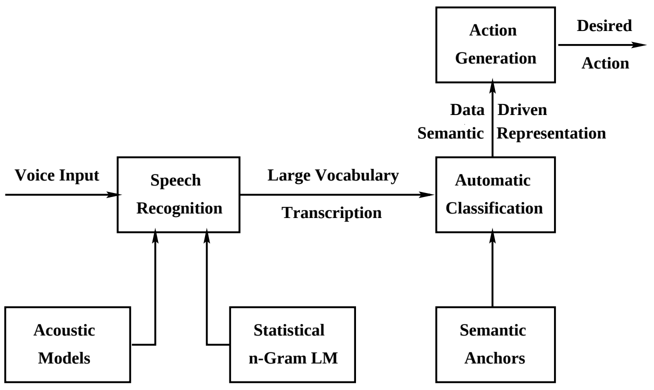

Fig. 2. Semantic Inference.

2. OVERVIEW

Conventional command and control is illustrated in Fig. 1. The typical system comprises three major components: a speech recognition engine, a mapping module, and an action generation engine. The role of speech recognition is to determine which command/query sequence (among those supported by the CFG) the user's voice input corresponds to. This sequence is then transformed to a semantic representation through some task-specific mapping rules. Obviously, these rules are closely tied to what can be parsed by the grammar. Finally, the semantic representation is converted to the desired action via the action generation engine.

The data-driven semantic inference framework is illustrated in Fig. 2. The CFG is replaced by a more general, statistical n-gram LM. The role of speech recognition is now to transcribe, hopefully with a reasonably low word error rate, the user's voice input into a sequence of words. In principle, there is no restriction on the formulation: the user can express the command in his or her own words. Accordingly, rule-based mapping is replaced by automatic semantic classification, whose role is to extract the data-driven semantic representation closest in meaning to the user's utterance. The intended action is then generated as previously.

Semantic classification relies on evidence presented in a training text database, disregarding the particular syntax used to express intended actions. It starts with the singular value decomposition of a matrix of co-occurrences between words and commands. Singular vectors can be interpreted as the representation of both words and commands in a continuous vector space of low dimension, called the LSA space. An important property of this space is that two command vectors which are "close" (in some suitable metric) tend to be associated with the same action, whether or not they are composed of similar word constructs. During training, each supported action gives rise to a different vector, referred to as the *semantic anchor* of the action. During recognition, mapping (the transcription of) a command onto an action amounts to comparing the corresponding command vector with each semantic anchor. The procedure is thus entirely data-driven.

3. LSA FRAMEWORK

Let $\mathcal{V}$ be a vocabulary of size M, and $\mathcal{T}$ a training corpus, i.e., a collection of variant wordings of commands to invoke N actions appropriate to some domain of interest. In this paper, each such wording, or *command variant*, is defined as a distinct sentence (such as "*what time is it?*" or "*what's the time?*") associated with a unique action (in this case, "*display current time*"). Typically, M and N are on the order of a few hundreds to a few thousands, and each command might have scores of different variations. The task at hand is to define a mapping between the set $\mathcal{T}$ and a vector space $\mathcal{S}$, whereby each command[1] in $\mathcal{T}$ is represented by a vector in $\mathcal{S}$.

We first construct a word-command matrix W associated with $\mathcal{V}$ and $\mathcal{T}$, in a manner similar to that detailed in [3]. This $(M \times N)$ matrix fully describes, for the training corpus $\mathcal{T}$, how often word w_i appeared in command d_j. We then perform the singular value decomposition (SVD) of W as:

$$W \approx \hat{W} = U\,S\,V^T\,, \tag{1}$$

where U is the $(M \times R)$ left singular matrix with row vectors u_i $(1 \leq i \leq M)$, S is the $(R \times R)$ diagonal matrix of singular values, V is the $(N \times R)$ right singular matrix with row vectors v_j $(1 \leq j \leq N)$, $R < \min(M, N)$ is the order of the decomposition, and ^{T} denotes matrix transposition. As is well known, $U^TU = V^TV = I_R$, the identity matrix of order R. Thus, the column vectors of U and V each define an orthornormal basis for the space of dimension R spanned by the u_i's and v_j's, i.e., the space $\mathcal{S}$ sought. The dimension R is bounded from above by the rank of the matrix W, and from below by the amount of distortion tolerable in the decomposition.

If we project the column vectors of W (i.e., commands) onto the orthonormal basis formed by the column vectors of U, the new coordinates are given by the columns of SV^T. This means that, for $1 \leq j \leq N$, the column vector Sv_j^T, or, equivalently, the row vector $v_j S$, characterizes the

[1]In contrast with previous work (cf. [6], [7]), our emphasis here is not on word representation, but on command representation. The underlying framework, however, is identical.

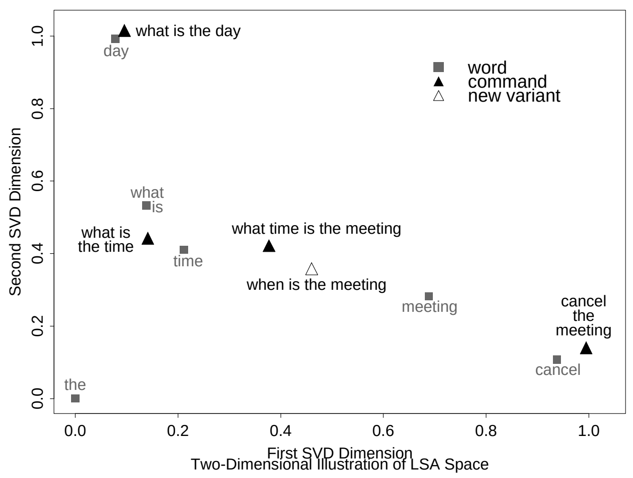

Fig. 3. Typical Behavior in Space $\mathcal{S}$ ($R = 2$).

position of command d_j in the space $\mathcal{S}$ [3]. Each of the N scaled vectors:

$$\bar{v}_j = v_j \, S = d_j^T \, U \, , \tag{2}$$

is therefore the semantic anchor of the action uniquely associated with the command $d_j \in \mathcal{T}$. Since $\hat{W}$ captures the major associational structure in W, the "closeness" of vectors in $\mathcal{S}$ is determined by the overall pattern of the language used in $\mathcal{T}$, as opposed to specific constructs. This tends to reveal meaningful associations (not just particular co-occurrences) between words and commands.

To illustrate, consider an application with $N = 4$ actions, each associated with a unique command: (i) "*what is the time*," (ii) "*what is the day*," (iii) "*what time is the meeting*," and (iv) "*cancel the meeting*." In this simple example, there are only $M = 7$ words in the vocabulary, with some interesting patterns: "*what*" and "*is*" always co-occur, "*the*" appears in all four commands, only (ii) and (iv) contain a unique word, and (i) is a proper subset of (iii). Constructing the (7×4) word-command matrix as described in [3], and performing the SVD, we obtain the 2-dimensional space shown in Fig. 3. This figure shows how each word and each command is representated in the space $\mathcal{S}$. Note that the two words which each uniquely identified a command—"*day*" for (ii) and "*cancel*" for (iv)—each have a high coordinate on a different axis, along with the respective command. Conversely, the word "*the*," which conveys no information about the identity of a command, is located at the origin. In addition, the word "*time*," which occurs in both (i) and (iii), indeed appears "close" to both.

4. SEMANTIC INFERENCE

This behavior means that a new variant, using a different wording, will still be "close" to the appropriate semantic anchor if that is otherwise consistent with the expected language structure (e.g., if the new variant would tend to express the corresponding desired action). Continuing with the example above, suppose that a user says something outside of the variants seen during training, such as "*when is the meeting*" rather than "*what time is the meeting.*" This new word string turns out to have a representation in the space $\mathcal{S}$ indicated by the hollow triangle in Fig. 3. Note that this point is closest to the representation of command (iii). Thus, the associated variant can be considered semantically equivalent to (iii), and the correct action can be automatically inferred.

To perform semantic classification, it is thus sufficient to determine, for a given variant, which semantic anchor the variant is most closely aligned with. This requires finding a representation for all individual variants in the space $\mathcal{S}$, including those not seen in the training corpus. Let us denote a variant by $\tilde{d}_p$, where the tilde symbol reflects the fact that this variant may not have been part of the training data. By construction, the feature vector $\tilde{d}_p$ is a column vector of dimension M, which can be treated as if it were an additional column of the matrix W. Provided the matrices U and S do not change, the SVD expansion (1) implies:

$$\tilde{d}_p = U \, S \, \tilde{v}_p^T \, , \tag{3}$$

where the R-dimensional vector $\tilde{v}_p^T$ acts as an additional column of the matrix V^T. Hence the *variant vector*:

$$\bar{\tilde{v}}_p = \tilde{v}_p \, S = \tilde{d}_p^T \, U \, , \tag{4}$$

corresponds to the representation of the variant $\tilde{d}_p$ in $\mathcal{S}$. Note that this representation is consistent with (2): if an action is expressed only with one variant, the resulting variant vector reduces to the semantic anchor for that action.

In the case of a variant not seen in the training corpus, relatively mild assumptions are required for (4) to hold. Clearly, if the new variant contains language patterns which are inconsistent with those extracted from W, the SVD expansion (1) will no longer apply. Similarly, if the addition of $\tilde{d}_p$ causes the major structural associations in W to shift in some substantial manner, U and S will no longer be valid, in which case it would be necessary to re-compute (1) to find a proper representation for $\tilde{d}_p$. If, however, the new variant generally conforms to the rest of $\mathcal{T}$, then $\bar{\tilde{v}}_p$ will be a reasonable representation for $\tilde{d}_p$.

Once (4) is obtained, it remains to evaluate the closeness between the observed variant vector $\bar{\tilde{v}}_p$ and the semantic anchors $\bar{v}_j$ determined during training. As it turns out, an appropriate closeness measure is the cosine of the angle between the two vectors [3]:

$$K(\bar{\tilde{v}}_p, \bar{v}_j) = \cos(\tilde{v}_p S, v_j S) = \frac{\tilde{v}_p \, S^2 \, v_j^T}{\|\tilde{v}_p S\| \, \|v_j S\|} \, , \tag{5}$$

for $1 \leq j \leq N$. Classification occurs by assigning the variant vector $\bar{\tilde{v}}_p$ to the semantic anchor $\bar{v}_j$ associated with maximum closeness.

Semantic inference has a number of useful properties. Not only is there no *a priori* constraint on what the user can say, but it can generalize well from a tractably small amount of training data. For example, it can learn synonyms and apply them in new contexts. Suppose for instance that the training data contained "*cancel the meeting*" and "*undo the* *meeting*" as variants of one command, but only "*cancel the last command*" as a variant of another command. Because the variants for the former command indicate that "*cancel*" and "*undo*" are synonyms, the new variant "*undo the last command*" would still be correctly mapped.

5. FRAMEWORK EXTENSION

Exploiting co-occurrences between words and commands, as done above, is an instance of the so-called "bag-of-words" model, which pays no attention to the order of words in the variants. While well-suited to capture large-span (semantic) relationships between words [6], [7], this makes it inherently unable to capitalize on the local constraints present in the language, be they syntactic or pragmatic. This is fine in applications where local constraints are discarded anyway (such as in information retrieval [4]), or for tasks such as call routing, where only the broad topic of a message is to be identified [5]. For general command and control tasks, however, this limitation may be more deleterious. Imagine two commands that differ only in the presence of the word "*not*" in a crucial place. The respective vector representations could conceivably be relatively close in the LSA space, and yet have vastly different intended consequences. Some commands may even differ only through word order, which makes them impossible to disambiguate.

We address this issue by extending the basic LSA framework via word agglomeration. The idea is to move from the characterization of co-occurrences between words and command to the characterization of co-occurrences between word n-tuples and commands, where each word n-tuple is the agglomeration of n successive words, and each command is now expressed in terms of all the word n-tuples it contains. The choice of n centers around the following trade-off: large enough to encapsulate meaningful local constraints, and small enough to preserve generalization properties and keep the problem computationally tractable. Agglomerations of $n = 2$ and $n = 3$ seem to work well. In the interest of brevity, details will be discussed elsewhere.

6. EXPERIMENTAL RESULTS

Experiments were conducted using the 113 actions currently supported in the "Speakable Items" desktop command and control task defined on MacOS 9. For each action, we collected from 5 to 10 distinct variants of the associated canonical command, from 10 different speakers. Eight test sets were considered, each comprising 113 commands expressed as similar (but not necessarily identical) variants collected from 2 other speakers, who had not seen the training variants. The objective was to use semantic inference to classify each test command variant against the appropriate action.

Three different setups were considered: (i) standard LSA, using simple word-command co-occurrences, i.e., word 1-tuples ($n = 1$); (ii) extended LSA, using word pair agglomeration ($n = 2$), and (iii) extended LSA, using word triplet agglomeration ($n = 3$). The resulting LSA matrices were (545 $\times$ 113), (2178 $\times$ 113), and (3720 $\times$ 113), respectively. In all cases we used $R = 62$ for the order of the decomposition. The results are reported in Table I, in terms of average classification error rate across all the test sets. These results suggest that semantic inference is largely viable in this kind of desktop command and control application.

Table I. Average Classification Error Rates.

Agglomeration Size	Classification Error Rate
$n = 1$	4.7 %
$n = 2$	2.5 %
$n = 3$	1.7 %

7. CONCLUSIONS

Semantic inference replaces the traditional rule-based mapping between utterance and action by a data-driven classification. This makes it possible to relax some interaction constraints. In particular, it obviates the need to specify rigid language constructs through a domain-specific (and typically hand-crafted) context-free grammar. This is turn allows users more flexibility in expressing the desired command/query, which tends to reduce the associated cognitive load and thereby enhance user satisfaction [5]. The underlying framework is latent semantic analysis. Each command variant is mapped onto an action by comparing the corresponding command vector with each semantic anchor, using the closeness measure (5) in the LSA space. To take advantage of local constraints, we extend the basic LSA framework using word agglomeration. Despite an increase in computational complexity, this extension is practical for semantic classification, because it tends to involve lower dimensions than large vocabulary recognition.

8. REFERENCES

[1] F. Jelinek, "Self–Organized Language Modeling for Speech Recognition," *Readings in Speech Recognition*, A. Waibel and K.F. Lee, Eds, Morgan Kaufmann, pp. 450–506, 1990.

[2] R. De Mori, "Recognizing and Using Knowledge Structures in Dialog Systems," in *Proc. Aut. Speech Recog. Understanding Workshop*, Keystone, CO, pp. 297–306, Dec. 1999.

[3] J.R. Bellegarda, "Exploiting Latent Semantic Information in Statistical Language Modeling," *Proc. IEEE*, *Spec. Issue Speech Recog. Understanding*, B.H. Juang and S. Furui, Eds, Vol. 88, No. 8, August 2000.

[4] S. Deerwester, *et al.*, "Indexing by Latent Semantic Analysis," *J. Am. Soc. Inform. Science*, Vol. 41, pp. 391–407, 1990.

[5] B. Carpenter and J. Chu–Carroll, "Natural Language Call Routing: A Robust, Self–Organized Approach," in *Proc. ICSLP*, Sydney, Australia, pp. 2059–2062, December 1998.

[6] J.R. Bellegarda, "A Multi-Span Language Modeling Framework for Large Vocabulary Speech Recognition," *IEEE Trans. Speech Audio Proc.*, Vol. 6, No. 5, pp. 456-467, September 1998.

[7] J.R. Bellegarda, "Large Vocabulary Speech Recognition With Multi-Span Statistical Language Models," *IEEE Trans. Speech Audio Proc.*, Vol. 8, No. 1, pp. 76-84, January 2000.