STATISTICAL SEMANTIC INTERPRETATION MODELING FOR SPOKEN LANGUAGE UNDERSTANDING WITH ENRICHED SEMANTIC FEATURES

Asli Celikyilmaz[(1)], *Dilek Hakkani-Tür*[(2)], *Gokhan Tur*[(2)]

[(1)]Microsoft Silicon Valley|[(2)]Microsoft Research
{asli|dilek|gokhan.tur}@ieee.org

ABSTRACT

In natural language human-machine statistical dialog systems, semantic interpretation is a key task typically performed following semantic parsing, and aims to extract canonical meaning representations of semantic components. In the literature, usually manually built rules are used for this task, even for implicitly mentioned non-named semantic components (like *genre* of a movie or *price range* of a restaurant). In this study, we present statistical methods for modeling interpretation, which can also benefit from semantic features extracted from large in-domain knowledge sources. We extract features from user utterances using a semantic parser and additional semantic features from textual sources (*online reviews, synopses, etc.*) using a novel tree clustering approach, to represent unstructured information that correspond to implicit semantic components related to targeted slots in the user's utterances. We evaluate our models on a virtual personal assistance system and demonstrate that our interpreter is effective in that it does not only improve the utterance interpretation in spoken dialog systems (reducing the interpretation error rate by 36% relative compared to a language model baseline), but also unveils hidden semantic units that are otherwise nearly impossible to extract from purely manual lexical features that are typically used in utterance interpretation.

Index Terms— semantic interpretation, spoken language understanding, graphical models, semi-supervised clustering.

1. INTRODUCTION

In a conversational understanding system, handling implicit and contextual information poses major challenges to current approaches for natural language (NL) utterance interpretation. For example in the entertainment domain, an utterance such as '*show me albums from the 80s*' may be interpreted solely based on the semantic frame '*the 80s*', (indicating the feature the *album-year*) to retrieve albums released between 1980 and 1990. However, user utterances often contain implied information. Given the utterance *"I wanna watch a movie that will make me laugh"*, a natural language understanding (NLU) engine, without an extensive knowledge base (KB) or labeled data, may fail to infer that the user is possibly requesting movies in the *comedy* genre. The goal of this paper is to present an utterance interpreter model, with specific application to utterances in which the semantic concepts are hidden or implied.

In a typical dialog system [1, 2, 3], a speech recognizer takes the user's spoken utterances and converts them into text. A NLU engine then extracts semantic information from the text, that will be analyzed by a dialog manager with knowledge of the domain, e.g., *travel*, *movies*, etc., to determine the next system action. The end-task is to retrieve and rank relevant information requested by the user (similar to question answering [4, 5], and summarization [6]).

Sample Dialog:

S_1: How can i help you?	
$\mathbf{U_1}$: *I want to watch tv-shows that will cheer me up?*	
S_2: Here are comedy shows playing on cable right now.	

Semantic Interpretation Module - SIM	
Inputs	**Utterance**: *I want to watch tv-shows that that will cheer me up?* **Domain Knowledge** : *movie synopses movie reviews*, entity lists (*movies, actors...*)
Features	**(I) NLU Features**:[semantic slots] slot(*type*) =*'tv-shows'*; slot(*descr.*)=*'cheer me up'* **(II) Tree Features**: (posterior probabilities) $P(U_1|$genre('*comedy*')) **(II) LM Features**: (likelihoods) *P*('*tv-show*'|genre('*comedy*'))
Outputs	**Facet**(Genre): comedy, romantic-comedy
Database Query-Q	select* from movies where genre in('comedy','romantic comedy')

Table 1. Depiction of the Semantic Interpretation Module (SIM) on an utterance U_1 from a sample dialog. The inputs, extracted features and outputs are attributes for the interpreter engine. Facets are used as attributes when forming standard queries $\boldsymbol{Q}$ by the domain expert $\boldsymbol{E}$.

Mainly hand-crafted rules or large dictionary look-up approaches are used to map semantic structures to attributes that can be queried in the database. In this paper, we propose a statistical modeling approach for learning interpretation rules and patterns. The new approach outperforms a baseline rule-based approach and is easier to adapt to other domains.

More specifically, we present a statistical semantic interpretation model (SIM) that classifies a given utterance into an in-domain semantic attribute (*genre, cuisine*), which can then be queried in the KB. For example, *"I'm in the mood for spicy food tonight"* can be classified as *cuisine*(*indian, chinese*). Due to the vast language variability in NL utterances, feature extraction for the interpretation task is intrinsically challenging. We initially extract from NL utterances several semantic components such as named entities (movie-name, time, or place), or other contextual semantic slots corresponding to genre, cuisine ('*vegetarian*') or nationality, etc. by way of a sequence classifier (see Table 1).

Recent research shows that exploiting lexical and semantic features in large unstructured text collections (e.g., online reviews, tweet conversations, blogs) can enhance tasks such as question answering [7], template based information extraction [8, 9], extracting semantic correspondence [10] as well as extracting facts, events or records [11], to name a few. In earlier work, several different probabilistic hierarchical clustering methods are proposed to extract information from unstructured text. A clustering method is suitable for our task, not only because the posterior probabilities from different classes may directly correspond to attributes that we wish to map utterances to, but also because clustering helps to unveil information hidden in vast amount of unlabeled text. We present a tree clustering model (extending the previous approaches above by tailoring for the interpretation task) to extract additional semantic features from unstructured text.

Our main contributions in this study are (***i***) construction of a statistical interpretation model to learn a mapping between a given utterance and a domain-specific attribute, (***ii***) a tree clustering model to extract domain specific semantic information from an unstructured text to unveil *implicit* semantic information patterns in utterances. Experiments are given in §4, in which we compare the performance of SIM to those of well-known probabilistic methods.

2. RELATED WORK AND MOTIVATION

In a dialog model, the NLU component consists of parsing a recognized spoken utterance into semantic structures and determining the meaning of its constituents (Table 1). Most work on NLU focus on issues arising from ambiguities in the lexicon (*'book'* can be noun or phrase), sense (*'bank'* has different meanings) and syntactic structure ambiguity ('*a flight to London at 9*' can have two possible readings). An interpreter engine plays a crucial role in translating the correct meaning representation from NLU components to generate a proper database query and retrieve relevant information.

A typical interpreter system is represented as a post-processing engine that uses expert-defined rules to translate NLU components into attributes [12, 13] that can be queried (in a database). For example, a context-independent normalizer translates extracted slots ("NY" $\rightarrow$"*New York*", "next Friday" $\rightarrow$"*12/14/2012*") based on domain-independent KB. In a more recent study, [14] use a rescoring strategy to interpret the output of multiple semantic classifiers to extract slots from NL utterances. Their results also include a rejection decision, but categorizing slot estimates within pre-defined types is not considered.

We approach interpretation as an utterance classification task to automatically map utterances onto specific attributes, a.k.a., facet types. A facet is an attribute of a domain. Existing web-pages, product descriptions or online collections of articles are usually augmented with navigational facets, e.g., *restaurant-type, price-range, genre*, etc. We take 'genre' facet in movies as a special case of semantic interpretation, though our method is generic enough to be used for other facets in a domain. The challenge of our task is to interpret utterances (map an utterance to a facet type) in which facet type is not always explicitly mentioned but can also be *implied*. In *'find scary movies'*, the token *'scary'* is tagged as movie-genre (by the semantic tagger), when the only related information in the domain knowledge source (i.e., database) is movie genre, the utterance is mapped to a genre type *'horror'* in movie domain. It is trivial to populate a dictionary of genre values for explicit genre mentions. However, when genre facets are implied, such as in *'find something that will cheer me up'*, we need an interpreter engine to map the utterance to *'comedy'* genre. To deal with these ambiguities, we introduce several rich semantic features and evaluate their effects on the interpreter engine in the experiments.

3. SEMANTIC INTERPRETER MODEL (SIM)

We build an utterance classifier model for semantic interpretation, to translate an utterance u into a semantic class corresponding to a genre facet $g \in G$, using the utterance level features extracted from NLU semantic parser (§3.1), the constraint tree clustering (CTC) similarity values per genre (§3.2), and language model (LM) likelihood features (§3.3) as presented below. We use a discriminative classification method, AdaBoost algorithm, a member of the boosting family of classifiers [15].

We use a corpus of NL user utterances (collected from human-machine dialogs such as in Table 1) and give guidelines to annotators to tag the utterances into several semantic components. Each utterance is tagged as one or more genre types by the interpreter engine as the output variable. In addition, the annotators assign semantic slots (tags) to each segment in utterances. Most of the slot values of an utterance are indicators of a domain specific facet. In this paper, we focus on ***genre facet*** types in the movies domain. Thus, slots mostly define named arguments of the movie related types, e.g., actor, director, movie-name, and are complemented by non-proper noun arguments, e.g., explicit genre (*"comedy"*), rating (*"PG13"*), duration (*"not very long"*). All other lexical units are tagged as 'other'. Additionally, we extract other slot types such as movie-description (*"terrifying"*) or movie-content (*"whose plane crashes"*). NLU engine detects slot types in new utterances, and the values of these slots are used as semantic features of the interpreter classifier.

3.1. NLU: Semantic Parsing Features

The first set of features for our interpreter, the SIM, is based on semantic parsing. A common approach to semantic parsing is to extract semantic slots corresponding to segments in an utterance via a sequence learning method. We take the slot filling model as a log-likelihood function $\mathcal{L}_p(\mathcal{D}_u;\beta_p)$ with training utterances $\mathcal{D}_u$=$\{u_1,..,u_N\}$ and feature weights β_p. We use Semi-Markov Conditional Random Fields (*Semi-CRF*) [16], a discriminative method for segmentation of sequences [17, 18]. Segmentation of an utterance represented as $\boldsymbol{s}$=$\{s_1,...,s_p\}$ where each segment s_j=$\langle b_j, e_j, y_j\rangle$ consists of a start b_j and end e_j position, and a label $y_j \in Y$. Let $\boldsymbol{f}$=$\langle g^1,...,g^T\rangle$ represent the vector of segment feature functions, each of which maps the pair (**u**,**s**) and an index j to a value $f^k(j,\mathbf{x},\mathbf{s}) \in \Re$ and $\mathbf{F}(\mathbf{x},\mathbf{s})$=$\sum_j^{|s|}\mathbf{f}(j,\mathbf{u},\mathbf{s})$. Semi-CRF use these feature functions in conjunction with the parameters β_p to represent the following conditional probability:

$$P(\mathbf{s}|\mathbf{u};\beta_p) = \frac{1}{Z(u)}e^{\beta_p\cdot\mathbf{F}(u,s)} \tag{1}$$

Our semantic parser uses two main feature sets:

- **Dictionary-based features:** For semantic tagging, these apply to features related to individual segments and their labels. Given a dictionary list, they specify which label is more likely. Using various sources such as web, manuals and domain specific product databases, we construct domain named-entity dictionaries, e.g., MOVIES, ACTORS, DIRECTORS, THEATERS, etc. and other entity dictionaries, e.g., GENRE, AWARDS. We also use domain independent dictionaries, e.g., CITY, DATE.
- **Pattern-based features:** We include noisy forms of patterns as constraints defined by a domain expert. These are in the form of regular expressions (e.g., "*directed by* {DIRECTOR}", "*stars in* {MOVIE}") as well as more complex rule-based grammars for patterns, e.g., DATE.

3.2. Features from Tree Clustering

The second set of features of the SIM is based on the similarity between a NL utterance and an unstructured text with a known attribute/facet type. These features indicate how likely a given utterance implies a certain genre. e.g., '*show me holiday*

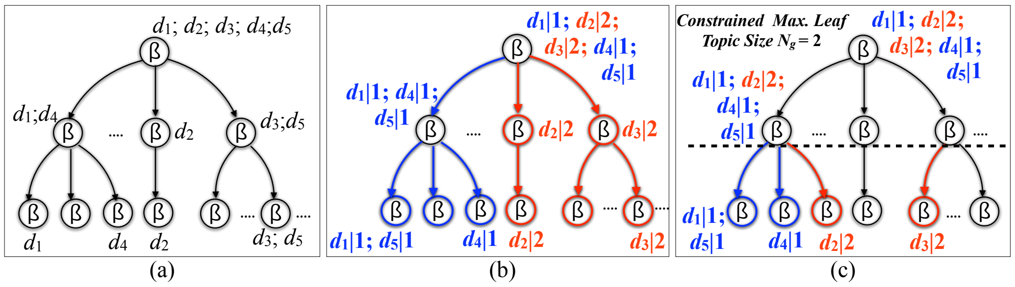

Fig. 1. (a) Unsupervised Tree Clustering; (b) Supervised Tree Clustering; (c) Constrained Tree Clustering (constraining the number of leaf topics (of a super topic) having the same label to $N_g = 2$.). Each node is a topic distribution over vocabulary. g_s={1, 2,..} indicate genre classes, where $d_2|2$ is a document with the genre label $g_2 = 2$.

movies'∼genre(*family*). Specifically, we use a similarity function between the movie reviews (each with genre label), and NL user utterances. The similarity function is defined on a hierarchical representation of concepts captured in movie reviews.

Recent research on information extraction from unstructured text [6, 19, 20, 21, 22] focus on the discovery of hierarchical concepts (from abstract to specific) in text documents using extensions of hierarchal topic models [23] and reflect this hierarchy on the sentences. Hierarchical concept learning models help to discover, for instance, that '*funny*' and '*hilarious*' are both contained in a general class '*comedy*', so that the utterances with specific terms can be related to more abstract concepts like comedies.

To learn the representations of unstructured text D such as reviews and synopses, we adopt the hierarchical topic model [23], which represents the distribution of topics in D by organizing them into a tree T of fixed depth L (Fig. 1): Each labeled document $\langle d_s|g_s\rangle$ pair [1] is assigned a path c={c^1,...,c^L} in the tree and each word w_i in a given document d_s is assigned a hidden topic z_c at level l of c. Each node is associated with a topic distribution over words parameterized by β.

In the standard form (Fig. 1.(a)), where no genre information is used at training time, the structure of labeled T is learned using a nested Chinese restaurant process (nCRP) [23], to define a distribution over words into paths in L-level tree T. In nCRP, assignments of labeled documents to paths are sampled sequentially: The first document d_1 takes the initial path, a single branching tree. Later, *m*th subsequent document is assigned to a path drawn from:

$$\begin{aligned} p(path_{\mathbf{old}}, c^l|m, m_c) &= \tfrac{m_c}{\gamma+m-1} \\ p(path_{\mathbf{new}}, c^l|m, m_c) &= \tfrac{\gamma}{\gamma+m-1} \end{aligned} \qquad (2)$$

where (m_c is the number of documents on the chosen path). Sampling from nCRP continues until the (L-1)th level is reached. Earlier work has introduced constraints on the structure of tree T mainly fixing the branch level L, where the set of topics associated with a document is known a *priori* [24, 25].

Similar to Labeled LDA [26], we use document genre labels to construct a labeled T. The application of such supervision is sketched on (Fig. 1.(b)). Specifically, each path gets the label of the documents assigned to them, e.g., the first document $d_1|1$ with genre=1 (as shown in blue) is sampled from the first path on the left and gets the blue color, and so on. However, such an approach may fail to capture the specific and abstract vocabulary attributing to a certain genre. Such a structure does not allow sharing of topics between different genre classes (paths on the figure) that potentially enable discovery of a compact genre specific vocabulary (topic clusters) at the leaf nodes. We demonstrate this with sharing of the vocabulary across topic clusters in Fig 3, which shows topic clusters of leaf nodes obtained from a supervised topic clustering of (Fig. 1.(b)). Note the two topic nodes, *comedy* and *family*, share many words, e.g., '*summer*', *adventure*', or '*kid-friendly*'. Thus, we need to constrain the supervision injected into the unsupervised hierarchical topic models to capture a vocabulary indicative of genre classes.

[1]e.g., g_s={1(*family*), 2(*horror*),..} takes on nominal values of different movie-genre classes, where $d_1|1$ indicates that document d_1 has the genre label $g_1 = 1$.

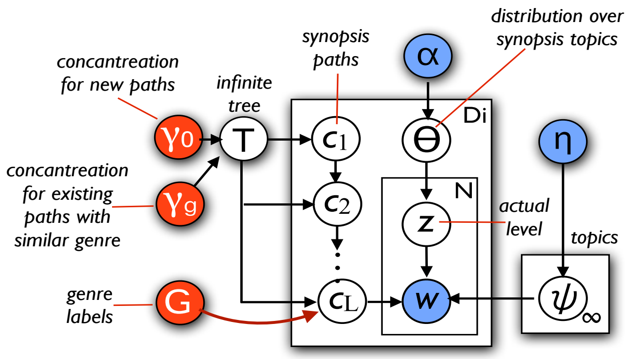
distribution over synopsis topics
synopsis paths
concantreation for new paths
infinite tree
concantreation for existing paths with similar genre
genre labels
actual level
topics
Di
N

Fig. 2. Depiction of Constrained Tree Clustering. **Blue** colored random variables are observed, **red** colored variables are introduced in our model as an extension to the standard topic clustering [27].

3.2.1. *Constrained Tree Clustering - CTC*

We present a new constrained tree clustering approach, as sketched in (Fig. 1.(c)). The assumption is that documents related to movies of different genre share common words, so we don't introduce supervision to the tree at the higher level nodes. Instead, at higher nodes, we let documents related to movies from different genre share common topics. At the (L-1)th level (one level above the leaf nodes), we use genre labels and constrain the branching strategy as follows: At the (L-1)th level, we allow each document d_s choose a path only from existing children of that node that has the same genre label (Fig. 1 (c)). The algorithm either samples from a new topic (and labels it g_s) or from one of the existing topics with the g_s label. Sampling from a new topic is limited by the maximum number of genre topics N_g (a user defined parameter) at the Lth level to increase the concentration of the same genre topics. The idea for limiting the number of genre

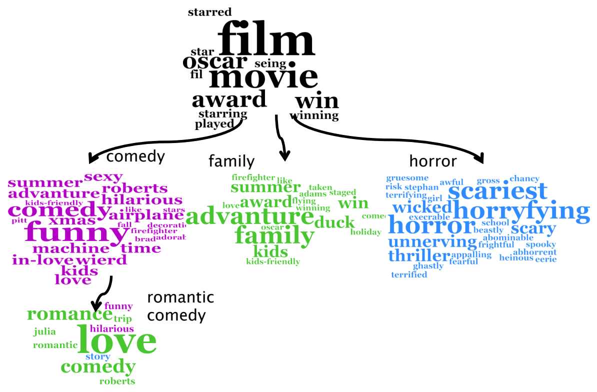

Fig. 3. A sample Supervised Tree Clustering of selected reviews. The topics on different paths have overlapping vocabulary, e.g., '*summer*', *advanture*', or '*kids*'.

topics is to enable sampling phrases of different utterances that infer the same genre (*"jump out of my skin"* and (movie) *"psycho"*) from the same topic. We influence the tree structure at this level by introducing two separate hyper-parameters for the nCRP prior. While sampling a document d_s with label g_s:

$\star$ if there are N_g number of existing leaf nodes with the same label as g_s, use $\gamma = \gamma_g$,

$\star$ otherwise, use $\gamma = \gamma_0$.

We keep an inventory of each labeled leaf node. If there is no child node with the same genre label, we sample from new paths until the number of nodes reaches N_g for each genre. We use a small value for γ_g ($0 < \gamma_g \lll 1$) by suppressing the generation of new branches for genres that already have enough child nodes. We let new documents have the option of creating new child nodes with probabilities proportional to γ_0 when there are not enough child nodes of that genre. We constrain the generation of new branches to a fixed number N_g by choosing $\gamma_g \lll \gamma_0$ and modifying the nCRP prior in Eq.(2).

The CTC generative process as depicted in Fig. 2 is shown in Algorithm 1.

Given a document d_s, θ_d is a vector of topic proportions from L dimensional $Dirichlet(\alpha)$ (distribution over levels in the tree). The nth word of d_g is sampled by first choosing a level $z_{d_s,n} = l$ from the θ_d with probability $\theta_{d,l}$.

3.2.2. *Features Obtained via CTC*

We use collapsed Gibbs sampling to learn CTC parameters. Given the assignment of words **w** to levels **z** and assignments of documents to paths **c**, the expected posterior probability of a particular word w at a given topic **z**=l of a path **c**=c is proportional to the number of times w was generated by that topic:

$$p(w|\mathbf{z},\mathbf{c},\mathbf{w},\eta) \propto n_{(\mathbf{z}=l,\mathbf{c}=c,\mathbf{w}=w)} + \eta \qquad (3)$$

Similarly, the posterior probability of a particular topic z in a given document d is proportional to the number of times z was generated by that document:

$$p(z|\mathbf{z},\mathbf{c},\alpha) \propto n_{(\mathbf{c}=c_{d_s},\mathbf{z}=l)} + \alpha \qquad (4)$$

where $n_{(.)}$ is the count of elements of an array satisfying the condition. After seeing the data, each path c gets multiple genre labels $c_g \in G$, and only the leaf nodes get a single genre label.

We use the posteriors of n-gram-topic and utterance-topic multinomials for each genre as additional features for the interpreter classifier as follows: Given an utterance u, and the tree, we find the

Algorithm 1. CTC Generation Process.

```
1. For each topic $k \in T$, sample a distribution
      $\beta_k \backsim$ Dirichlet($\eta$).
2. For each level $l \in${2,..,($L$-1)}
      $*$ draw a topic from node $c^{l-1}$ using nCRP($\gamma_0$)
3. At genre level $l$=$L$, draw a topic
      $*$ from $c_g^{l-1}$ using nCRP($\gamma_g$) if $N_g$ nodes exist,
      $*$ else from node $c_g^{l-1}$ using nCRP($\gamma_0$)
4. For each labeled document $\langle d_g, G_g \rangle \in D_g$,
      $(a)$ Sample $L$-vector $\theta_d$ mixing weights from
         Dirichlet distribution $\theta_d \sim Dir(\alpha)$.
      $(b)$ For each word $n$, choose: $(i)$ level $z_{d,n}|\theta_d$
         and $(ii)$ word $w_{d,n}|\{z_{d,n}, c_d, \beta\}$
```

*red colored lines (2 and 3) describe the new nCRP prior.

similarity of the user uttrerance u to the closest path for each genre c_g based on word-topic and topic-document posteriors:

$$sim(g|u) = sim_\phi(c_g|u) * sim_\theta(c_g|u) \qquad (5)$$

Let $\bar{p}_{c_g,l}$ be a sparse probability distribution of node l of path c_g over all the words from all the documents on the path (from same genre), averaged by the number of documents N_d on the path. Let $p_{u,l}$ be a sparse distribution of node l of path c_g over the posteriors of all the words in utterance u. $sim_\phi(c_g|u)$ is a measure of divergence of $p_{u,l}$ from $\bar{p}_{c_g,l}$

$$sim_\phi(c_g|u) = \frac{1}{L}\sum_{l=1}^{L} 10^{-IR(\bar{p}_{c_g,l},p_{u,l})} * l \qquad (6)$$

Words that are more specific to utterances get higher probabilities at the child nodes (e.g., in *"show me movies that others find terrifying"*, "*terrifying*" is a more specific word than *"show")* . Thus, we boost the similarity, which is based on the Kullback-Leibler (KL) measure[2], as the level is closer to the leaves. The second measure is based on the divergence between the utterance-topic mixing distributions $p_u = p(\mathbf{z}_{u_i}|c_g)$ and $p_{c_g} = \frac{1}{N_d}p(\mathbf{z}_{d_g}|c_g)$ (Eq. 4), where N_d is the number of documents in the path c_g:

$$sim_\theta(c_g|u) = 10^{-IR(p_{u_i},p_{c_g})} \qquad (7)$$

We use $sim(g|u)$ in Eq. (5) to represent CTC features per genre.

3.3. Language Modeling Features

Language modeling has been shown to be a powerful paradigm for information retrieval applications (Lafferty and Zhai, 2003; Zhai, 2008, [29, 30]) in terms of representing the concepts and language of the domain. They first postulate a model for each document and for a given query select the document that is most likely to have generated the query. In contrast to earlier work, we build supervised language models on sets of documents, which are pre-labeled with a specific genre, and later obtain likelihood features for each utterance. We encode supervision in the form of manually constructing clusters D_s of reviews/synopses from the same *genre* g_s and building a separate genre language model, LM_g, for each cluster. Given D_s we train n-gram language models ($\mathtt{LM}_g$) (n<4) in which the probability of a given string **w** for each genre g (such as semantic structure (slot) value or full utterance) is given by:

$$p(\mathbf{w}|g) = \prod_{i=1}^{|q|} p(w_i|w_{i-1},...,w_{i-(n-1)}) \qquad (8)$$

where $|q|$ is the length of the string. We build $\mathtt{LM}_g$s on the reviews and synopsis clusters separately. Then the usual Bayesian formulation is applied for inference:

$$\hat{g} = \arg\max_g p(g|\mathbf{w}) = \arg\max_g p(\mathbf{w}|g) \times p(g) \qquad (9)$$

[2]IR(p,q)=KL(p$||\frac{p+q}{2}$)+KL(q$||\frac{p+q}{2}$) [28]

where $p(g)$ is the prior probability for genre g.

We trained LM_gs using SRILM [31], with Kneser-Ney smoothing and using the default parameters. In the experiments, we show several n-gram posteriors per `LM` model that are used as additional constraints to the interpreter classifier.

4. EXPERIMENTS AND DISCUSSIONS

Here we present the results of our experiments based on the performance of the interpreter engine that uses the various feature sets presented in this paper. We discuss the effects of using estimated semantic components in comparison to truth values (actual components) as well as features from unstructured text.

Data. Our corpus consists of 10K utterances labeled with 19 unique genre classes, e.g., animation, family, classics, comedy, etc. Some utterances contain no genre information and get 'n/a' tag. In addition to output genre tags, each utterance is labeled with 41 unique slot tags attributed to named entities and phrases in utterances, e.g., movie/actor/director name, description, language, etc. We downloaded around 80K synopses from Netflix.com and 200K online movie reviews from IMDB.com.

Features. The interpreter features are as follows:

- ***f-lexical***: N-gram features up to n=3.
- ***f-slot*-E**: Binary feature (per slot type) indicating whether a segment in an utterance is tagged with that slot type. There is one feature per slot type E.
- ***f-slot*-V**: Slot value features and has the value of the segment if the segment in the utterance is tagged with a slot type. One feature per slot type.
- ***f*-LM**: LM likelihoods as features obtained from genre specific LMs per slot values. For instance, given "[*black and white*]$_{movie-type}$ *movies*", we obtain the likelihood per genre LM given movie-type slot as such: *p*("*black and white*"|LM$_{comedy}$), *p*("*black and white*"|LM$_{classics}$), etc.
- ***f*-CTC**: Features representing the similarity of an utterance to each genre class, as measured through the constrained tree T, using Eq. (5). We extract one similarity value per genre.

Model. We analyze the performance of the interpreter module using the genre labels as output values and compute classification error rate by comparing system estimated labels with manually annotated ones. In each experiment we use five-fold cross validation, where each time a fold is left out for measuring the model's performance. The final measure is the average performance of each fold. At `CTC` training time, we include a set of training utterances with the same genre as additional documents. To limit the amount of paths specific to each genre, we use dual-hyper-parameters and choose a small value $\gamma_g = 10e^{-4} \ll \gamma_0$ for when the max-number of path per genre is reached. We experimented with various depth tree structures $l = 3, 4, 5$ and found that $l = 3$ is the optimal depth.

4.1. Analysis on the Semantic Interpreter

We present interpretation as an utterance classification task, using several constraining features to build a robust model. We analyze the performance of our interpreter engine against several baselines:

1. Dictionary-based (**`DIC`**) baseline deterministically selects a genre label given an utterance, as defined in §3.1.
2. The Language Model (**`LM`**) baseline uses semantic components (slots) within utterances to capture per genre likelihoods. Among the NLU slots, *genre* and *description* slots specifically correspond to *explicit* genre mentions. Thus, we use these slots to extract probabilities e.g., $p(s_{genre}|LM_g)$ and calculate the conditional likelihood of the user utterance given each genre g to find the optimum genre g^*:
$$g^* = \arg\max_g p(g|s_{genre}, s_{description}) \propto p(s_{genre}|LM_g) * p(s_{description}|LM_g)$$
3. Naive Bayes (**`NB`**): Given LM posterior probabilities on lexical units in an utterance u, and certain slot values, we make a "naive" conditional independence assumption, and compute conditional distribution over genre g labels:
$$p(g|u, S) \propto \quad p(g) \prod_{i=1}^{|q|} p(w|LM_g) * p(s_{genre}|LM_g) * p(s_{description}|LM_g)$$
where S=$\{s_{genre}$,$s_{description}\}$, is the list of slot values, $p(g)$ is the genre class prior distribution obtained from the training data and $|q|$ is the number of lexical units in utterance u. For an observed u, an optimum genre $g*$ is chosen if the class conditional is maximized $g^* = \arg\max_g p(g|u, S)$.

To investigate the performance of our interpreter against the baselines above, we build various `SIM` models using different features:

4. LM in `SIM` (**`SIM-LM`**), uses f-LM features to build interpreter classifier,
5. N-gram lexical features in `SIM` (**`SIM-Lex`**), uses only n-gram features *f-lexical*,
6. LM and lexical features in `SIM` (**`SIM-LM-Lex`**),
7. CTC in `SIM` (**`SIM-CTC`**), uses only constrained similarity scores per genre from CTC (Eq. 5),
8. CTC with lexicals in `SIM` (**`SIM-CTC-Lex`**), uses CTC similarity scores per genre with lexicals.

	Model Type	Interpretation Error Rate
Baselines	DIC	13.1%
	LM	6.7%
	NB	7.8%
	SIM-Lex	8.9%
	SIM-LM	5.6%
	SIM-LM-Lex	**4.3%**
	SIM-CTC	**5.7%**
	SIM+CTC+Lex	**4.2%**

Table 2. Benchmark semantic interpreter models. **`DIC`**: Dictionary Lookup (Baseline); **`LM`**: Language Model; **`NB`**: Naive Bayes; **`SIM-`**$: Our semantic interpterer with features **`Lex`**: n-grams, **`LM`**- likelihoods from genre specific LMs, **`CTC`**: posteriors from tree clustering.

The performance results are shown in Table 2. The `SIM` models achieve the best performance, significantly reducing the error rate 89%, 37%, and 46% relative compared to the dictionary `DIC`, `LM`, and `NB` baselines consecutively. Furthermore, with the `SIM` models, using features from unstructured text can actually help to capture hidden (implied) genre facets. `CTC` can discover hidden concepts in synopses or reviews that correspond to implied semantic structures in NL utterances. Thus, the clustering approaches help to discover hidden information from unstructured text, which can then be used as additional features for the semantic interpreter engine. A simple classifier based on lexical features may not be able to capture the implicit genre information hidden in the NL utterances.

4.2. Analysis on Extracted Features

Here we analyze the contribution of each set of features on two seperate interpretation models: (1) `SIM-bin` a binary interpretation model which detects if there is a genre facet information in a given utterance ($g = 1$) or not ($g = 0$) (the genre is neither implied nor explicitly mentioned in the utterance), and (2) `Sim-mult` a multi-way classifier that maps utterances into one of the genre facet labels.

Features	SIM-bin		SIM-mult	
	Actual	Est.	Actual	Est.
(1) *f-lexical*	4.8%	–	5.9%	–
(2) *f*-LM	3.9%	–	5.6%	–
(3) *f*-CTC	4.7%	–	5.7%	–
(1)+*f-slot*-E	3.5%	4.0%	5.3%	6.2%
(1)+*f-slot*-V	3.2%	3.9%	5.1%	6.0%
(1)+(2)	**2.6%**	–	4.3%	–
(1)+(3)	3.5%	–	4.2%	–
(1)+(2)+(3)+ *f-slot*-E + *f-slot*-V	**2.2%**	–	**3.9%**	–

Table 3. Interpretation error rates of five-fold cross-validation for different features and methods. Bold values indicate statistical significance over baseline.

We predict a given utterance's semantic structure (slots) using NLU semantic parsing explained in §3.1. The predicted slots are used as features for the interpreter classifier. To analyze the effect of the NLU semantic parser's performance on the interpreter, we compiled additional sets of features using the actual slot values of the interpreter features. We denote the interpreter models that use actual slot labels (features *f-slot*-E and *f-slot*-E) as ***Actual***, whereas, the predicted slot labels are denoted as ***Estimates (Est.)***.

The experiment results are shown in Table 3. The addition of almost every feature set improves the performance of the model, though the increase in performance is not large. For instance the F-LM feature corresponding to *movie-director* slot helps to correctly estimate the utterances such as *"movies like directed by james cameron"*, or *"Hitchcock type movies"*. However there are only handful of these types of utterances in our dataset. Thus, addition of posterior probabilities from unstructured text as constraints on top of n-gram features improves the robustness of the interpreter classifier. Using all the features extracted from unstructured text outperforms the rest of the models. In addition, the interpreter is slightly affected by the NLU prediction errors (Actual vs. Est.). The error rate reduces by 1-2% absolute in average.

Using only the lexical features (F-Lex) provides a good baseline performance. The reason for this is that, it is not uncommon for utterances to have explicit semantic frames that can be easily mapped/estimated to correct genre facets. It is the implied genre utterances where the lexical models are not strong at predicting the correct genre. Using features extracted from unstructured text, (2) and (3) in Table 3, improves the lexical model performance on predicting the implied genre utterances.

5. CONCLUSIONS

In this paper, we presented a classifier based semantic interpreter model for translating NL utterances into pre-defined facets. We demonstrated that implementation of constrained clustering tools to extract hidden concepts from unstructured text can improve interpretation of user utterances, specifically when the utterances contain implied concepts. In future work, we'll investigate the extension of our interpreter module to facets in different domains.

6. REFERENCES

[1] P. Price, "Evaluation of spoken language system: the atis domain," *DARPA Speech and Natural Language Workshop*, 1990.

[2] M. McTear, "Spoken dialogue technology," *Springer*, 2004.

[3] G. Tur and R. De Mori, "Spoken language understanding: Systems for extracting semantic information from speech," *Wiley*, 2011.

[4] M. Wang, N.A. Smith, and T. Mitamura, "What is the jeopardy model? a quasi-synchronous grammar for qa," *Proc. EMNLP*, 2007.

[5] D. Racichandran and E. Hovy, "Learning surface text patterns for a question answering system," *Proc. ACL*, 2002.

[6] R. Barzilay and L. Lee, "Catching the drift: Probabilistic content models with application to generation and summarization," *Proc. HLT-NAACL*, 2004.

[7] S. Cucerzan and E. Agichtein, "Factoid question answering over unstructured and structured context on the web," *Proc. The Text Retrieval Conference (TREC)*, 2005.

[8] N. Chambers and D. Jurafsky, "Template-based information extraction without the templates," *Proc. of ACL*, 2011.

[9] D. Feng, G. Burns, and e. Hovy, "Extracting data records from unstructured biomedical full text," *Proc. of EMNLP*, 2007.

[10] P. Liang, M. I. Jordan, and D. Klein, "Learning semantic correspondance with less supervision," *Proc. of ACL*, 2009.

[11] E. Benson, A. Haghighi, and R. Barzilay, "Event discovery in social media feeds," *Proc. of ACL*, 2011.

[12] S. Seneff, L. Hirschman, and VW. Zue, "Interative problem solving and dialoge in the atis domain," *Proc. 4th DARPA Speech and Natural Language Workshop*, 1991.

[13] J. Wright, A. Gorin, A. Abella, and J. W. Allen, "Spoken language understanding within dialogs using a graphical model of task," *Proc. of ICSLP*, 1998.

[14] C. Raymond, F. Bechet, N. Camelin, R. De Mori, and D. Damnati, "Sequential decision strategies for machine interpretation of speech," *IEEE Trans. Audio, Speech and Language Processing*, vol. 15, 2007.

[15] R. E. Schapire and Y. Singer, "A boosting-based system for text categorization," *Machine Learning, vol. 39. no 2/3*, vol. 39, pp. 135–168, 2000.

[16] S. Sarawagi and W.W. Cohen, "Semi markov conditional random fields for information extraction," *Proc. of NIPS*, 2004.

[17] X. Li, "Understanding the semantic structure of noun phrase queries," *Proc. of ACL*, 2010.

[18] S. Singh, D. Hillard, and C. Leggetter, "Minimally-supervised extraction of entities from text advertisements," *Proc. NAACL*, 2010.

[19] H. Daume and D. Marcu, "Bayesian query-focused summarization," *Proc. ACL 2006*, 2006.

[20] J. Eisenstein and R. Barzilay, "Bayesian unsupervised topic segmentation," *Proc. EMNLP-SIGDAT*, 2008.

[21] D. Wang, S. Zhu, T. Li, and Y. Gong, "Multi-document summarization using sentence-based topic models," *Proc. ACL 2009*, 2009.

[22] A. Haghighi and L. Vanderwende, "Exploring content models for multi-document summarization," *NAACL HLT-09*, 2009.

[23] D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum, "Hierarchical topic models and the nested chinese restaurant process," *Proc. of NIPS*, 2003.

[24] J. Reisinger and M. Paşca, "Latent variable models of concept-attribute attachement," *Proc. of ACL*, 2009.

[25] Y. Petinot, K. McKeown, and K. Thadani, "A hierarchical mdoel of web summaries," *Proc. of ACL*, 2011.

[26] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," *Proc. EMNLP*, 2009.

[27] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, 2003.

[28] C. Manning and H. Schuetze, "Foundations of statistical natural language processing," *In MIT Press. Cambridge, MA*, 1999.

[29] W. Dakka, "Automatic discovery of useful facet terms," *Proc. SIGIR Faceted Searach Workshop*, 2006.

[30] D. Downey, S. Schoenmackers, and O. Etzioni, "Sparse information extraction: Unsupervised language models to the rescue," *Proc. ACL*, 2007.

[31] A. Stolcke, "Srilm - an extensible language modeling toolkit," *Proc. Intl. Conf. on Spoken Language Processing*, 2002.