Interspeech 2018
2-6 September 2018, Hyderabad

Semantic Lattice Processing in Contextual Automatic Speech Recognition for Google Assistant

Leonid Velikovich, Ian Williams, Justin Scheiner, Petar Aleksic, Pedro Moreno, Michael Riley

Google, Inc

leonidv@google.com, iantw@google.com, jmscheiner@google.com, apetar@google.com, pedro@google.com, riley@google.com

Abstract

Recent interest in intelligent assistants has increased demand for Automatic Speech Recognition (ASR) systems that can utilize contextual information to adapt to the user's preferences or the current device state. For example, a user might be more likely to refer to their favorite songs when giving a "music playing" command or request to watch a movie starring a particular favorite actor when giving a "movie playing" command. Similarly, when a device is in a "music playing" state, a user is more likely to give volume control commands.

In this paper, we explore using semantic information inside the ASR word lattice by employing Named Entity Recognition (NER) to identify and boost contextually relevant paths in order to improve speech recognition accuracy. We use broad semantic classes comprising millions of entities, such as songs and musical artists, to tag relevant semantic entities in the lattice. We show that our method reduces Word Error Rate (WER) by 12.0% relative on a Google Assistant "media playing" commands test set, while not affecting WER on a test set containing commands unrelated to media.

Index Terms: speech recognition, contextual speech recognition, named entity recognition

1. Introduction

A contextual ASR system uses real-time signals from the speaker's context to improve recognition accuracy. Bringing context into ASR bridges the gap between a recognizer which models an average speaker in normal conditions and one that is adapted to personal and situational context. While many techniques are available to take advantage of context, we focus here on language model (LM) rescoring. Rescoring adjusts LM probabilities in real time based on contextual signals, and allows targeted adjustments without the need for training a context-specific LM. Work has been done on incorporating a variety of contextual signals into ASR: the device type being used, the history of the speaker's queries or actions, device state, location and dialog state, among others [1, 2, 3].

In this paper, our contextual ASR system uses a two-pass LVCSR (large-vocabulary continuous speech recognition) recognizer containing a first-pass decoder similar to the system described in [4] with a first pass LM and a more accurate second-pass rescoring language model LM. The ASR system can use n-gram models or neural networks, our only requirement is that the decoder outputs the set of possible hypotheses in the form of a word lattice, which is an efficient weighted automaton representation of speech hypotheses [5]. We report result on the system that uses n-gram models.

Various rescoring systems have been described in the literature. They can use weighted contextually relevant n-grams [6, 7] and can contain contexually added dynamic classes [8]. Such systems perform on-the-fly n-grams rescoring during first-pass decoding [9]. Using semantic information for contextual ASR is a topic that has attracted recent interest [10].

In this paper, we describe a new method for bringing semantic contextual information into ASR. Our method identifies semantically relevant entities and phrases in the word lattice and provides a mechanism for rescoring them.

Our proposed semantic rescoring system can effectively annotate (tag) and rescore millions of semantic entities. For example, we can rescore broad collections such as the set of all known songs including international and obscure songs, as well as contractions or colloquial ways of referring to songs. We refer to these collections as semantic classes. We use a Semantic Lattice Tagger [11] to annotate semantic classes directly in the ASR word lattice using a modification of techniques from Speech Information Extraction and Semantic Interpretation [12, 13, 14, 15]. Our approach allows restricting tagging to just a subset of contextually relevant semantic classes and entities. After tagging, thousands of n-gram contexts may be used to rescore each semantic class, resulting in a rescoring system that supports billions of unique n-grams in real-time. Our approach uses FSTs (Finite-State Transducers) and works with readily available libraries such as OpenFST [16] and OpenGRM [17].

The remainder of this paper is organized as follows. Section 2 provides an overview of the system's components and how they interact. Section 3 provides a description of the individual components: how the semantic tagging model is trained (3.1) and applied (3.2), Section 4 describes how rescoring is performed on a word lattice containing semantic tags. Experimental results are described in Section 5 and conclusions in Section 6.

2. System Design

In this section we describe the main components of the contextual ASR system that uses semantic lattice rescoring.

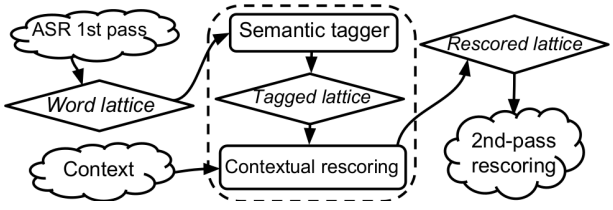

Figure 1: *Semantic Lattice Rescoring Contextual ASR system.*

As shown in Figure 1, ASR first-pass decoding outputs a word lattice, an acyclic weighted graph over words that efficiently encodes proposed hypotheses of the user's utterance.

10.21437/Interspeech.2018-2453

Taking this lattice as input, the Semantic Lattice Tagger inserts open/close decorator tags around likely contextually relevant semantic entities in the lattice. Each inserted decorator pair denotes a semantic class (such as <song> </song> or <musical_artist> </musical_artist>). Multiple (ambiguous) semantic interpretations are allowed and expressed as parallel paths through the same words, surrounded by different decorator tags.

The next step is rescoring of the semantically-tagged lattice based on user-specific context. The hypotheses that contain a meaningful phrase involving an anticipated class of semantic entities are rewarded. For example, a speech recognition task intended for playing music could look for <song> or <musical_artist> and rescore "play $SONG" or "play $SONG by $MUSICAL_ARTIST". This would improve recognition quality in scenarios where such inputs are more likely than average. Similar to the semantic tagging step, rescoring must be done efficiently on the entire lattice. With semantic rescoring applied, the modified lattice is passed to second-pass rescoring used by the recognizer. As a second-pass rescorer is not expected to know about semantic decorator tags, they can be either removed or temporarily hidden.

3. Semantic Tagger

3.1. Training

This section explains how the Named Entity Recognition (NER) model used by the Semantic Lattice Tagger is trained. The complete algorithm is described in [11] and a summary provided here.

We start with an existing large-scale structured knowledge model, albeit whose latency and memory requirements make it impractical for real-time use in ASR. Treating this model as an oracle that can annotate training sentences in bulk, we distill the model into a much smaller FST statistical n-gram model that fits in memory on a single machine (see Figure 2).

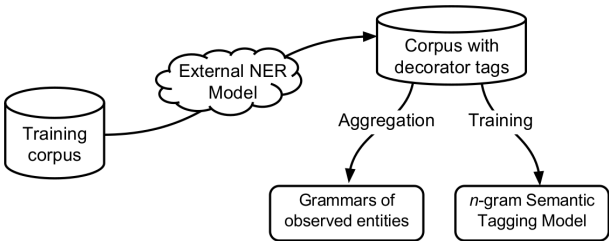

Figure 2: *A system for training the semantic lattice tagger model.*

The resulting n-gram language model represents the probability of a sentence containing optional decorator tags, for example:

```
how tall is <alternative_rock_artist> taylor
swift </alternative_rock_artist> ?
```

Transition costs on each arc represent negative log probabilities as described in [5]. The model represents the joint probability of seeing the sentence together with the tags, i.e., $P(s,t)$, where s is a sentence and t is a choice of zero or more tags representing a distinct semantic interpretation of s.

While processing the training data, we also build a histogram of observed semantic entities sorted by their frequency over the training corpus. Examples of captured entities are shown in Table 1. The long tail is rich with international entities, contractions, alternative wordings, alternative or incorrect spellings, and obscure entities; which can be valuable to the recognizer. In the next section, we show how this histogram is used to control the search space of semantic interpretations.

Percentile	Sample Class	Sample Instance
1st	tv_series	game of thrones
10th	tv_program	CBB UK
25th	actor	aiysha hart
50th	composer	stefan djuric
90th	film	billy and mandy boogey adventure

Table 1: *Examples of media-related semantic entities from the head to the long tail of the frequency distribution. Stefan Đurić is a Serbian musician; the name of the animated film Billy & Mandy's Big Boogey Adventure is contracted in the last example.*

3.2. Lattice Tagging

At recognition time, the Semantic Lattice Tagger inserts decorator tags into an untagged ASR word lattice. As an acyclic graph, the word lattice can represent exponentially many hypotheses relative to its size; its richness makes it desirable to perform NER directly on the lattice using FST operations rather than extracting n-best hypotheses to process individually.

The tagging algorithm is described in more detail in [11]; however a brief summary is provided below.

Introduce tags: Given the word lattice L with weights removed, identify every possible entity in it using an FST grammar of the top-N million entities observed during model training.

$$L_{\text{expanded}} \leftarrow \text{rmweight}(L) \circ C \tag{1}$$

Where the $\circ$ operator denotes FST composition. The grammar constrainer FST C expands the lattice, injecting balanced left and right decorator tags around any matching entities. As a result, parallel, non-nested paths are created for ambiguous entities that may belong to multiple classes (or to no class).

Note that we can restrict our generation to just a relevant subset of classes and/or entities by having task-matched grammar constrainers at our disposal. This reduces our search space and allows fine-grained control over what gets rescored or doesn't.

Rank the taggings: Assign our proposed tags conditional probabilities $P(t|s)$, so that we can find the most likely tagging for any word sequence in the lattice. Note that our n-gram model is not a conditional but rather a joint probability model, P(s, t). We calculate the conditional probability over an input lattice by using Bayes' rule:

$$P(t|s) = \frac{P(s,t)}{P(s)} \tag{2}$$

We get $P(s,t)$ by composing the expanded lattice from step (1) with our n-gram model FST:

$$L_{\text{joint}}(s,t) \leftarrow L_{\text{expanded}} \circ P(s,t) \tag{3}$$

We get the marginal P(s) by deleting the decorator tags from $P(s,t)$ and then determinizing and minimizing the lattice to sum probabilities of paths through the same words:

$$P_{\text{marginal}}(s \in L) = \sum_t L_{\text{joint}}(s,t) \quad (4)$$

Finally, we perform division by the marginal $P(s)$ as a multiplication (i.e., FST composition) by its reciprocal. The reciprocal of P(s) is trivially computed by reversing the sign of its negative-log probability arc weights:

$$P(t|s) \leftarrow P_{\text{marginal}}(s)^{-1} \circ L_{\text{joint}}(s,t) \quad (5)$$

Optimize: max-normalization is a way of dealing with probability fragmentation between possibly many semantic interpretation of the same words. It is performed by scaling up tagging probabilities, so the optimal tagging(s) of every hypothesis has a modified probability P'=1. This effectively degenerates probabilities into penalty costs for non-optimal taggings; this technique works well in practice because ASR's goal is to pick the best hypothesis regardless of tags. This approach was successfully used in ASR pronunciation modeling [18, 19]). Pruning is also performed, deleting taggings that trail too far behind the optimal tagging to avoid clutter. This is described in [11].

Lastly, compose the conditionally tagged tattice with the original ASR lattice, such that each resulting hypothesis' probability is the product of its first-pass decoder probability and the max-normalized conditional tagging probability:

$$L_{\text{combined}} = P_T(t|s) \cdot P_{\text{1st}}(s) \quad (6)$$

Note that overall strategy described here (expanding a word lattice using an entities grammar and then ranking them with a statistical model) has been successfully used for Speech Information Extraction and Semantic Interpretation tasks [12, 13, 14, 15]. What makes our approach different is the goal (contextual ASR accuracy) and algorithms implemented to support that goal: computing conditional probabilities, performing max-normalization, and modeling annotations as standalone decorator words instead of as tags attached to words. This avoids dramatic increase of vocabulary size when scaling to hundreds of classes and millions of entities, and allows some semantic modeling efficiencies.

4. Rescoring Using Semantic Tags

After decorator tags with probabilities are introduced into the lattice, the next step is to rescore using the tags. Based on the use case, a set of n-grams that include nonterminals referring to semantic classes are provided. For example, a music-playing application may provide `play $SONG`. When a lattice path matches one of the n-grams, we apply a cost adjustment to that path. We must also determine where and by how much to adjust lattice arc scores to achieve good quality results.

We use a simple approach to adjust the path costs when one of the n-grams is encountered in the lattice. We subtract a fixed amount of negative log-probabilty (equivalent to increasing the likelihood) on the arc containing the open decorator tag, so that the path cost through that phrase is reduced. For our initial experiments we optimized this score reduction over a dev set. In future applications we intend to use automated learning of rescoring models, as described in [7].

Similar to semantic tagging, an efficient approach requires operating at the word lattice level, therefore we use FST composition for this task. This composition has a custom implementation; on the FST encoding the n-grams to rescore, we introduce two labels φ and ρ. The φ label indicates that the arc is followed whenever there is a failure to match any other arc leaving the state. When an arc with φ is followed, nothing is consumed in the lattice and nothing is emitted from the composition. Similarly, ρ is also followed when no other arc matches, but in this case the lattice arc is consumed and the composition emits the lattice arc unchanged. When the complete n-gram including the semantic entity has been matched, we apply the score boost, α. All other arcs leave the score unchanged. Encoding the example n-grams `play $SONG`, and `play some $COMPOSER` as an FST looks like figure 3.

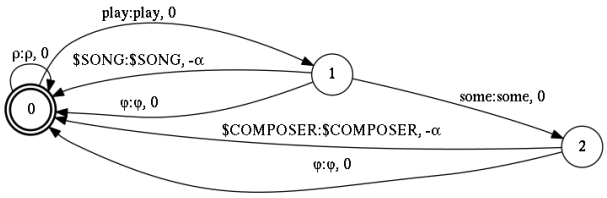

Figure 3: *A simple rescoring model for semantic entities.*

There is one remaining issue to solve, which is to match the decorator tag span from the lattice to the non-terminal form in the rescoring n-grams. E.g., the lattice would contain the label sequence `play <SONG> thriller </SONG>`, and we need to match `play $SONG`. To achieve this, we wrap the arc matching logic on the rescoring FST to map the decorator tag to the non-terminal before matching the label. The remaining lattice arcs until the end of the decorator span are emitted unmodified. Table 2 shows how each lattice label would be looked up, and what the resulting score change would be.

Lattice label	play	<SONG>	thriller	</SONG>	now
Match	play	$SONG	{none}	{none}	now
Output / boost	play / 0	<SONG> / α	thriller / 0	</SONG> / 0	now / 0

Table 2: *Example matching between lattice FST arcs and rescoring FST arcs*

In practice we implement all of these customizations using the OpenFst [16] library, making use of composition filters and FST arc matchers. A complete discussion of this is outside the scope of this paper, but the interested reader can refer to [20].

5. Experimental Results

In this section, we evaluate the contextual ASR system that uses semantic lattice tagging and rescoring. The system is evaluated on the test set consisting of transcribed utterances and on live Google Assistant traffic. Note that a detailed evaluation of the Semantic Lattice Tagger in isolation, including its faithfulness to the oracle semantic knowledge model and its tagging latency, is included in [11]. In this paper, we focus on end-to-end performance in a production ASR environment.

5.1. Evaluation on Transcribed Test Sets

We evaluate performance on two test sets, a test set consisting of media playing commands (containing 1,289 utterances) and a test set consisting of non-media queries (conatining 2,204 utterances). 88 patterns corresponding to audio entities were

used for rescoring, such as "`play $MUSICAL_ARTIST $SONG`", each containing one or more of 11 semantic classes listed in Table 3.

`film`	`composition`	`creative_work`
`song`	`entertainer`	`musical_album`
`composer`	`tv_program`	`musical_artist`
`concerto`		`fictional_character`

Table 3: *Semantic classes used in rescoring patterns.*

Hypotheses in the tagged ASR lattice matching rescoring patterns were rescored by -3.0 in negative-log space, a value that was optimized over a held-out set containing both in-domain and out-of-domain utterances. Results are shown in Table 4. We saw improved accuracy on the play media test set while no regressions were observed on the non-media test set. Regressions due to false positive semantic interpretations were rare due to the fact that we only rescore a semantic entity when matching a non-trivial pattern.

Test set	Number of utterances	WER [%]	WER (resc.) [%]	Change [%]
Media	1,289	5.00	4.40	-12.0
Not media	2,204	8.80	8.80	0.0
Combined	3,493	7.40	7.18	-3.0

Table 4: *Test set with Assistant media and non-media utterances.*

Table 5 illustrates the kinds of wins we saw on the Media test set. Although one might intuitively expect most wins to look like example (a), replacing a non-entity with an entity, this was often not the case. We also found that the right combination of consecutive entities was assigned better scores than nonsensically arranged entities, and even words adjacent to an entity were also corrected. This was made possible by the flexibility of using rescoring patterns rather than rescoring entities directly.

(a) Non-entity replaced by a semantic entity:	
Base:	play moriah carey
Exp.:	play <musical_artist> mariah carey </musical_artist>
Ref.:	Play Mariah Carey.
(b) An improbable entity sequence $SONG $COMPOSITION is replaced with a valid combination $SONG $ARTIST:	
Base:	play play <song> oh me </song> <composition> cheerleader </composition>
Exp.:	play <musical_artist> omi </musical_artist> <song> cheerleader </song>
Ref.:	Play OMI Cheerleader.
(c) Adjacent word "play" is corrected by being rescored together with the semantic entity:	
Base:	the <musical_artist> lady gaga </musical_artist>
Exp.:	play <musical_artist> lady gaga </musical_artist>
Ref.:	Play Lady Gaga.

Table 5: *Example wins on media playing utterances when rescoring the priors for matched patterns by -3.0 in negative-log space.*

5.2. Side-by-side Comparison on Google Assistant Traffic

We also performed quality evaluation experiments on live Google Assistant traffic. We used an evaluation method referred to as a side-by-side (SxS). In a SxS, a baseline ASR system is run in parallel with an experimental ASR system. The baseline does not perform semantic lattice processing and rescoring, while the experimental side performs it but is otherwise identical. Live traffic voice commands are recognized using both systems, and output differences are collected for human rating. All data is anonymized and and personally-identifiable information is removed.

The raters determine whether a transcript change is an improvement, a regression, or neither, termed "win", "loss", and "neutral", respectively. An improvement does not imply a perfect transcript or even an improvement in WER; it indicates that a human would consider the results more meaningful. Similarly, two transcripts that are judged the same can be different in content but yield equally useful transcriptions. To maintain a blind experiment, the raters are not aware of the specific nature of the task. Additionally, the baseline and experiment sides are presented to the raters in a random order so they cannot know which is which.

In the SxS experiments, 648 rescoring patterns were used, containing one or more of three visual media-related classes (`actor`, `film`, `tv_program`) were rescored by -2.0.

Diff Size	Wins	Losses	Neutral
440	78	27	335

Table 6: *Side-by-side experiments analysis of utterances that resulted in differences.*

The SxS rating results are presented in Table 6. The experimental system produces 2.89 win-loss ratio over the baseline system due to semantic lattice processing and rescoring.

While aggregating utterances used in the SxS experiments, we recorded median input sizes (the number of states in the ASR word lattice) and the median latencies of Semantic Lattice Tagging and Semantic Rescoring. The results are shown in Table 7; in the average case, the semantic system added less than 2.4 ms to the total recognizer latency. Our observed 1.8ms latency for tagging is slightly lower than the 2.8ms previously reported in [11]; this is explained in part by the difference in domain traffic and partly by the smaller number of semantic classes used in the current experiment (3 vs. 300 classes used in [11]).

Median ASR lattice size	14.15 states
Median tagging latency	1.839 ms
Median rescoring latency	0.520 ms

Table 7: *Utterance statistics in SxS experiments.*

6. Conclusion

In this paper, we described a contextual ASR system that uses semantic lattice processing and rescoring in order to improve recognition accuracy. We showed that our system reduces WER by 12% relative on media playing commands test set as well as brings quality gains on live Google Assistant traffic. We analyized a use case in which a broad group of semantic entities such as songs or films are used in lattice processing. Directions for future work include expanding to more domains, automatically learning weighted rescoring patterns, and using stronger contextual signals.

7. References

[1] P. Aleksic, C. Allauzen, D. Elson, A. K. D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *ICASSP 2015*, 2015.

[2] J. Scheiner, I. Williams, and P. Aleksic, "Voice search language model adaptation using contextual information," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 253–257.

[3] A. Michaely, J. Scheiner, M. Ghodsi, P. Aleksic, and Z. Wu, "Unsupervised context learning for speech recognition," in *Spoken Language Technology (SLT) Workshop*, 2016.

[4] G. Saon, D. Povey, and G. Zweig, "Anatomy of an extremely fast lvcsr decoder," in *Proc. Interspeech*, 2005, pp. 549–552.

[5] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[6] P. S. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. B. Hall, B. Roark, D. Rybach, and P. J. Moreno, "Bringing contextual information to google speech recognition." in *Proc. Interspeech*, 2015, pp. 468–472.

[7] A. H. Michaely, M. Ghodsi, Z. Wu, J. Scheiner, and P. Aleksic, "Unsupervised context learning for speech recognition," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 447–453.

[8] L. Vasserman, B. Haynor, and P. Aleksic, "Contextual language model adaptation using dynamic classes," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 441–446.

[9] K. B. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Interspeech 2015*, 2015.

[10] W. Zhu, "Using knowledge graph and search query click logs in statistical language model for speech recognition," in *Proc. Interspeech 2017*, 2017, pp. 2735–2738. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2017-1790

[11] L. Velikovich, "Semantic model for fast tagging of word lattices," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 398–405.

[12] B. Favre, F. Béchet, and P. Nocéra, "Robust named entity extraction from large spoken archives," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2005, pp. 491–498.

[13] C. Raymond, F. Bechet, R. De Mori, and G. Damnati, "On the use of finite state transducers for semantic interpretation," *Speech Communication*, vol. 48, no. 3, pp. 288–304, 2006.

[14] C. Servan, C. Raymond, F. Béchet, and P. Nocéra, "Conceptual decoding from word lattices: application to the spoken dialogue corpus media," in *The Ninth International Conference on Spoken Language Processing (Interspeech 2006-ICSLP)*, 2006.

[15] S. Hahn, M. Dinarelli, C. Raymond, F. Lefevre, P. Lehnen, R. De Mori, A. Moschitti, H. Ney, and G. Riccardi, "Comparing stochastic approaches to spoken language understanding in multiple languages," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1569–1583, 2011.

[16] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, ser. Lecture Notes in Computer Science, vol. 4783. Springer, 2007, pp. 11–23, `http://www.openfst.org`.

[17] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The opengrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, 2012, pp. 61–66.

[18] G. Chen, H. Xu, M. Wu, D. Povey, and S. Khudanpur, "Pronunciation and silence probability modeling for asr," in *Proceedings of INTERSPEECH*, 2015.

[19] E. Fosler, M. Weintraub, S. Wegmann, Y.-H. Kao, S. Khudanpur, C. Galles, and M. Saraclar, "Automatic learning of word pronunciation from data," in *the Proceedings of the International Conference on Spoken Language Processing*, 1996.

[20] C. Allauzen, M. Riley, and J. Schalkwyk, "Filters for efficient composition of weighted finite-state transducers," in *International Conference on Implementation and Application of Automata*. Springer, 2010, pp. 28–38.