SEMANTIC MODEL FOR FAST TAGGING OF WORD LATTICES

Leonid Velikovich

Google Inc., New York, NY

leonidv@google.com

ABSTRACT

This paper introduces a semantic tagger that inserts tags into a word lattice, such as one produced by a real-time large-vocabulary speech recognition system. Benefits of such a tagger include the ability to rescore speech recognition hypotheses based on this metadata, as well as providing rich annotations to clients downstream. We focus on the domain of spoken search queries and voice commands, which can be useful for building an intelligent assistant. We explore a method to distill a pre-existing very large named entity disambiguation (NED) model into a lightweight tagger. This is accomplished by constructing a joint distribution of tagged n-grams from a supervised training corpus, then deriving a conditional distribution for a given lattice. With 300 tagging categories, the tagger achieves a precision of 88.2% and recall of 93.1% on 1-best paths in speech recognition lattices with 2.8ms median latency.

Index Terms— word lattice, named entity disambiguation, knowledge distillation, lattice rescoring

1. INTRODUCTION

Substantial prior work has been done on tagging semantic entities in sentences [1] and spoken utterances, including spoken term detection (STD) [2]. Many approaches focus on explicit parsing, however, such approaches may have trouble dealing with ungrammatical input common in errorful speech recognition [3] or choppy language used in search queries. More direct approaches bypass syntactic analysis and use machine learning to identify or disambiguate semantic entities [1, 3]. More recently, the availability of large-scale supervised knowledge resources, including Wikipedia and Freebase, stimulated further research of named entity recognition (NER) and disambiguation [4, 5, 6, 7, 8]. The increasing accuracy yet heavy resource utilization of such systems makes it tempting to use them as a source of labels on which to train a smaller, supervised tagging model, with the goal of low latency and high accuracy over a target distribution. One possible application is large-vocabulary continuous speech recognition (LVCSR) first-pass lattice rescoring, where recognition hypotheses from a speech decoding first pass are rescored using more fine-grained methods [9]. For example, one may wish to bias recognition towards certain categories of entities (e.g., restaurants) based on the dialog state from which the utterance was spoken [10]. Some accuracy and performance improvements were found by performing speech recognition and semantic tagging concurrently [11, 12].

Generally, discriminative methods such as conditional random fields (CRF) [13] are preferred over generative approaches for tagging [14, 15]; however, they have their drawbacks, including difficulty scaling to a very large number of states [16], slow approach of asymptotic error rate [14], and restricting input to a standalone sentence rather than a word lattice. Work has been done on adapting discriminative models to a finite-state transducer (FST) framework, making them available to use on word lattices [17]. Such approaches look promising, although current implementations can be memory- and CPU-intensive.

We present a method that instead takes a generative approach, modeling a joint distribution of n-grams with semantic tags and later deriving a conditional distribution for a given lattice. Compared to existing methods, the first benefit of our approach is the ability to start with a very large model (or annotated training corpus) and scale it down to an arbitrarily small size and fast tagging speed, making it practical for real-time speech recognition. The second benefit is our model's handling of ambiguity. In the scenario of automatic speech recognition (ASR) word lattice rescoring, tagging a highly ambiguous sentence would split it into numerous alternate tagged paths, each one improbable in an absolute sense; this fragmentation hurts the entire hypothesis' chance of being chosen over a non-ambiguous hypothesis. We address this by supporting "max-normalization", where probabilities are scaled such that the best tagging gets a "probability" of 1, preserving n-best order. We also support beam pruning of improbable taggings to avoid cluttering the lattice. Our approach uses FSTs and works with readily available libraries such as OpenFST [18] and OpenGRM [19].

Note that the approach of expanding the lattice with possible taggings and ranking them using a statistical model has already proven successful on smaller datasets, including work on extracting semantic interpretations [20, 21, 22]. We focus on scalability to billions of training instances and millions of entities, and adaptation to speech recognition via beam pruning and max-normalization.

The remainder of the paper is organized as follows. Section 2 describes training the distilled model by choosing a set of classes, preparing training data, and building a joint distribution. Section 3 describes the lattice tagging algorithm, which involves converting the joint distribution into a conditional distribution for a given lattice. Section 4 describes our experimental results with accuracy and latency numbers, and Section 5 is the conclusion.

2. TRAINING THE TAGGING MODEL

We started with a pre-existing very large (over 1TB) discriminative NED model, which can be thought of as a black box that can annotate large volumes of training data in an offline, batched manner. It would also be possible to start with a large corpus annotated through other means, such as (ideally) human-introduced tags. We assume that the resulting annotated corpus represents our best available knowledge of the truth.

The NED model supports more than 10,000 semantic collections (classes), organized into hierarchical relationships with multiple inheritance. For the purpose of building a fast and compact tagger, we reduced these to a flat set of 300 classes by agglomerating sibling collections into their parent. This was based on (1) cosine similarity between members of a collection and the entire collection using a context window of 3 tokens to the left and right; (2) in-class entropy, dropping collections heavily dominated by a few members; and (3) frequency over an annotated sample.

We started with a corpus of ~200 billion search and voice command queries, and annotated it using the NED model. Semantic collection annotations were mapped to our 300 classes, and opening and closing decorators were inserted around entities, e.g.:

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

The decorator tags were then treated as regular words, and a Katz backoff n-gram language model (LM) was trained on the annotated corpus. Before training, numeric entities such as telephone numbers and currencies were aggregated into nonterminal symbols for generality. The LM was entropy-pruned to a target size of 100 million n-grams and stored as an FST, with transition costs on each arc representing negative log probabilities as described in [23].

This LM models the joint distribution $P_T(s,t)$, where the subscript T denotes a LM trained on data with decorator tags, s is a sentence, and t represents a particular way of tagging s.

3. LATTICE TAGGING ALGORITHM

An astute reader may observe that we can simply use $P_T(s,t)$ as the LM in a first-pass speech recognizer, which would automatically generate a word lattice with semantic tags in it. Unfortunately, this has proven impractical, as non-determinism in $P_T(s,t)$ introduces search errors, word accuracy reduction, and higher CPU utilization. Instead, we focus on tagging performed as a postprocessing step on an input lattice L.

The essence of our algorithm is to use Bayes' rule to derive the conditional distribution $P_T(t|s)$ from the joint $P_T(s,t)$, accomplished by dividing the joint by the marginal probability:

$$P_T(t|s) = \frac{P_T(s,t)}{P_T(s)} \tag{1}$$

We compute each of the above terms for an input lattice using FSTs. Our input includes:

- Input lattice L: an acyclic, weighted automaton over words.
- A transducer to insert decorator tags into L.
- The joint distribution $P_T(s,t)$.

For best performance, L is first determinized (with pruning to control the size of the output) and minimized. For example, if no path can be rescored in the second pass by $\geq \delta$ weight, we can prune all hypotheses that are $\geq \delta$ weight behind the best path:

$$L_{Opt} \leftarrow min(det_{weight=\delta}(L)) \tag{2}$$

3.1. Inserting tags into the lattice

To insert decorator tags into L_{Opt}, we built a grammatical "constrainer", C. C is an unweighted transducer whose input side accepts all untagged sentences and output side accepts optionally tagged sentences, subject to: (1) balanced open-close tags; (2) no nested tags; (3) text inside the tags corresponds to class instances observed in training data (see Figure 1). C is constructed with a single master state that loops on any word in the language (represented by a wildcard label, σ) and on any observed instance of a permitted class, surrounded by open/close decorators.

We found that constraining the lattice in this manner not only discards bad taggings, it greatly reduces our search space and gives finer control over latency. Composing unweighted L_{Opt} with C yields an unweighted lattice representing all possible taggings of L (see Figures 2 and 3 for an example):

$$L_T \leftarrow rmweight(L_{Opt}) \circ C \tag{3}$$

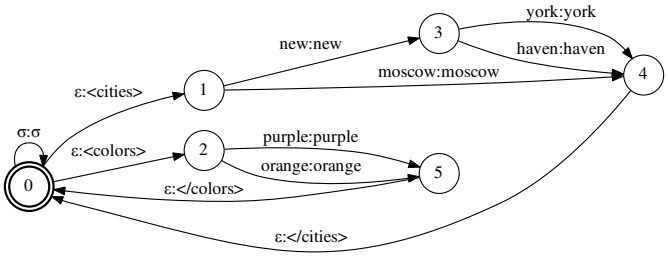

Fig. 1. Simple grammar constrainer C.

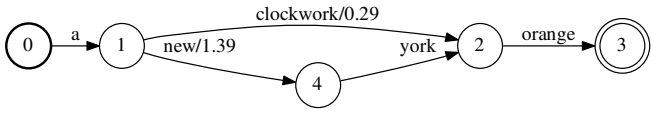

Fig. 2. Toy lattice L. Arc weights are -log probabilities; no weight represents probability one.

3.2. Computing the joint probability

We fetch probabilities from the joint distribution $P_T(s,t)$ by composing:

$$L_J(s,t) \leftarrow L_T \circ P_T(s,t) \tag{4}$$

Note that P_T, being a back-off LM, contains failure transitions denoted by the label ϕ that are followed when there is no match at the current FST state [23]. A composition algorithm must be used that supports this functionality, such as [18]. Figure 4 shows L_J, which is lattice L decorated with all possible taggings and whose arc weights are joint probabilities from $P_T(s,t)$.

3.3. Computing the marginal probability

Now we can compute the marginal probability, $P_T(s)$. We know of no algorithm to compute $P_T(s)$ precisely from a cyclic automaton such as the entire LM $P_T(s,t)$; the twins property makes this a non-trivial problem [24, 25]. However, it suffices to compute the marginal from just the (acyclic) lattice L:

$$P_T(s \in L) = \sum_t P_T(s,t) \tag{5}$$

To compute $P_T(s \in L)$, we remove decorator tags from the output arcs of L_J by projecting on input, remove epsilons, and determinize, combining all paths through the same words using the weighted determinization algorithm which is guaranteed to terminate on acyclic inputs [24]:

$$L_M(s) \leftarrow min(det_{weight=\delta}(rmeps(\uparrow L_J))) \tag{6}$$

3.4. Max-normalization

In the case of ASR, standard probability normalization may be undesirable because the probability of a hypothesis divided among several equiprobable taggings is unfairly handicapped w.r.t. a hypothesis with just one tagging during Viterbi decoding (a similar problem has been observed in speech pronunciation modeling [26, 27]). Max-normalization, where probabilities are scaled such that the highest "probability" becomes 1, has proven useful in such situations [26, 27]. We can enable max-normalization by having min and det operators use arc costs in the tropical semiring, where probability addition is replaced with taking the maximum [23]. The computed quantities are no longer true probabilities and will be referred to as pseudo-probabilities. As Figure 5 illustrates, instead of representing the sum of joint probabilities of all taggings of a sentence, the modified L'_M represents joint probability of the best tagging of each sentence:

$$L'_M(s) = \max_t P_T(s,t) \tag{7}$$

3.5. Computing the conditional probability

We can now divide the joint distribution L_J by the marginal L_M to get the conditional. Division is performed by inverting arc weights in the denominator and composing the numerator with it:

$$L_C(t|s) \leftarrow min(det_{wt.=\delta'}(inv(L_M(s)^{-1} \circ L_J(s,t)))), \tag{8}$$

where inv denotes inversion of input and output arc labels and the superscript -1 denotes inverting arc weights. When using max-normalization, we are actually normalizing by the cost of the best path as computed in (6), giving the best path a conditional pseudo-probability of 1 (see Figure 6):

We can prune taggings more than δ' beam behind the optimal tagging(s) during the determinization step in (8).

3.6. Applying the conditional probability

Now that we have the conditional distribution, we can apply it to the original (optimized) word lattice:

$$L_{Combined} \leftarrow L_C(t|s) \circ L_{Opt} \tag{9}$$

$L_{Combined}$ contains the original weights of L modified by added tags (Figure 7). When using max-normalization, paths with optimal taggings keep the original path probability in the lattice, preserving their n-best order.

In speech recognition, weights in the original lattice L encode the probability $P_{1st}(s)$, i.e., the probability assigned to any path s by the first-pass LM, so $L_{Combined}$ is equivalent to:

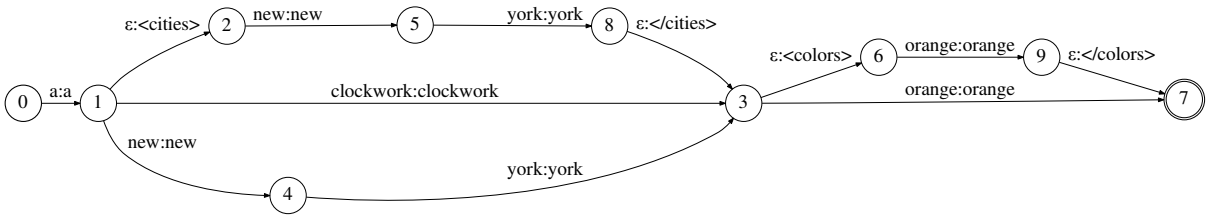

Fig. 3. Expanded lattice L_T shows all ways of tagging L.

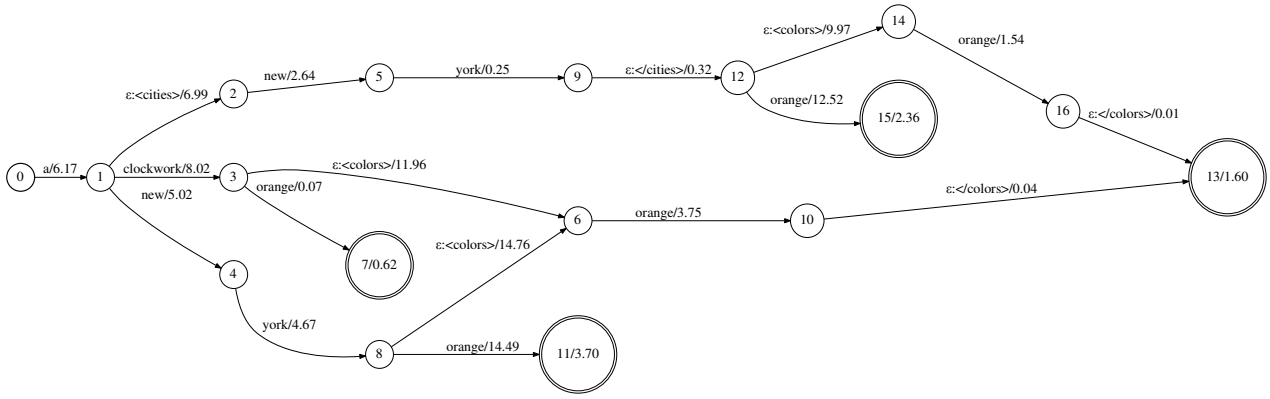

Fig. 4. L_J, expanded lattice with the joint probabilities. For readability, arcs that have the same input and output label are displayed with only one label, e.g., "a:a" is displayed as "a".

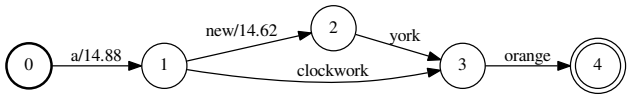

Fig. 5. Marginal pseudo-probabilities over the lattice L'_M.

$$L_{Combined} = P_T(t|s) \cdot P_{1st}(s) \tag{10}$$

$P_{1st}(s)$ is a more efficient sentence-level model than $P_T(s)$ because it does not contain tags. Thus we are combining a more accurate sentence-level model with our best available tagging knowledge.

4. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the lattice tagger in terms of accuracy and latency percentiles.

Our test set is a random sample of 12,879 voice search utterances for which the speech recognizer produced first-pass lattices (i.e., they were not rejected as noise). The utterances were anonymized and cleaned of personally identifiable information.

4.1. Quality measurements

For measuring quality, the lattice 1-best, 10-best, and human transcript were taken for each utterance. The truth set was constructed by annotating each sentence with the pre-existing NED model. The class tag statistics are shown below:

Data	# sentences	% with tags	Tags/sentence
1-best	12,818	65.9	1.040
10-best	128,326	63.4	0.996
Human	12,879	65.5	1.043

We ran the lattice tagger on each sentence and checked for agreement between each (non-decorator) word's classification and its "truth" classification. A confusion matrix was used to calculate precision P, recall R, and F_1 measure as a function of true or false positives and negatives:

$$P = \frac{|T.P.|}{|T.P. \cup F.P.|} \tag{11}$$

$$R = \frac{|T.P.|}{|T.P. \cup F.N.|} \tag{12}$$

$$F_1 = \frac{2 \cdot P \cdot R}{(P + R)} \tag{13}$$

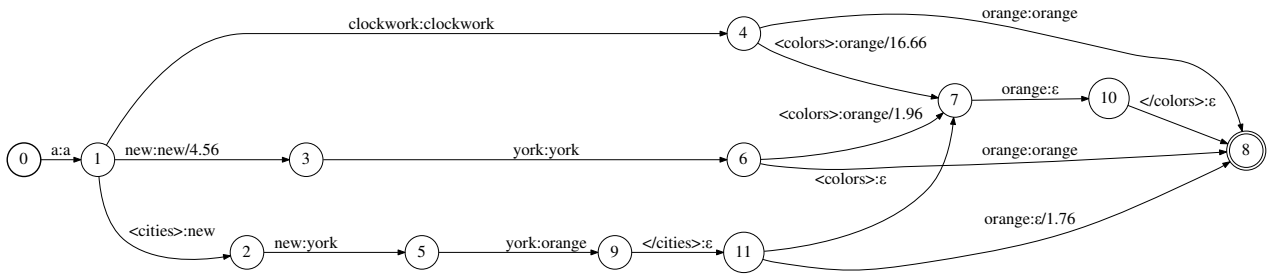

Fig. 6. Lattice L_C with conditional pseudo-probabilities. Note that the best tagging of "a new york orange" is "a $\langle cities \rangle$ new york $\langle /cities \rangle$ $\langle colors \rangle$ orange $\langle /colors \rangle$", while the best tagging of "a clockwork orange" has no tags, as there are no movie title tags in our example constraining grammar. The best tagging of any sentence has a log cost of 0.

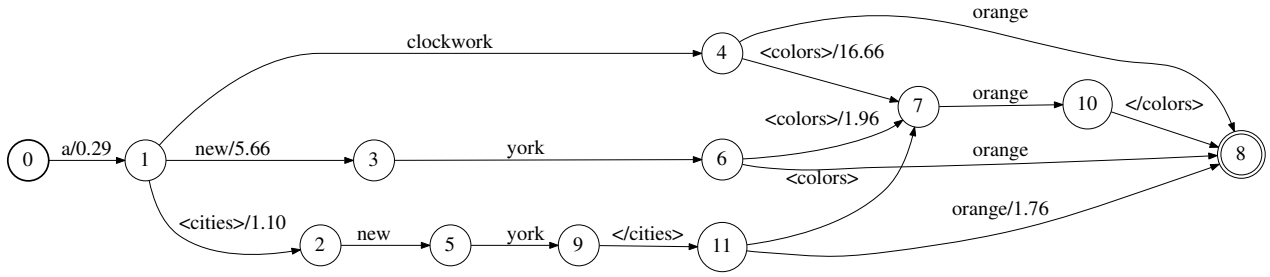

Fig. 7. Lattice L_C with tagged sentence probabilities, projected on the input side. The best path through any word sequence has the same probability as in the original lattice.

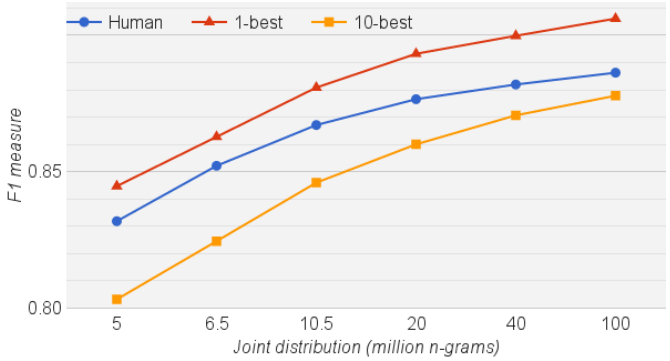

Fig. 8. Joint distribution P_T size vs. F_1 measure.

Figure 8 shows F_1 numbers as a function of the joint distribution model P's size. The constrainer C was kept at its maximum size of 6 million entities (~100MB).

The data shows that a larger, more accurate joint distribution P_T gives better precision and recall. Note that we used the relative entropy pruning method from [28] to shrink P_T. While effective for language modeling, this method is likely suboptimal for a tagging model, as not all of the relative entropy in P_T states is relevant for tagging decisions. (Exploring better ways to compress P_T is a topic of future exploration.)

In our experiments, 1-best hypotheses showed the highest agreement; we believe this is because (a) they are more similar to the kind of training data used to build the joint distribution and constraining grammar; (b) the lower-probability hypotheses in 10-best are of lower quality and less prone to annotator agreement due to contextual ambiguity. Metrics for the operating point of 100 million n-grams (2.3GB size) are:

Data	Precision [%]	Recall [%]	F_1 measure
1-best	88.2	93.1	0.906
10-best	85.5	90.2	0.878
Human	85.4	92.1	0.886

Figure 9 shows recall numbers as a function of the size of the constrainer C (with the joint distribution P_T fixed at 100 million n-grams). We don't graph precision here because it remains virtually constant across C sizes. This is probably because a larger C offers additional tagging options, but the ranking of those options is still performed by the joint distribution P_T.

While the constrainer has a small memory footprint (100MB for 6 million entities and 13MB for 0.54 million entities), we found that reducing its size helps curtail search space and manage latency, as becomes apparent in the section below.

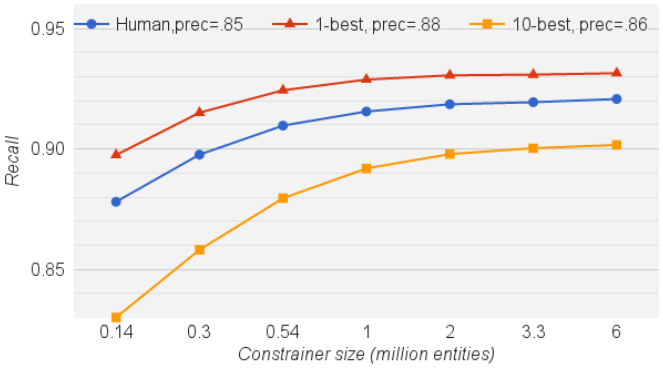

Fig. 9. Constrainer C size vs. recall.

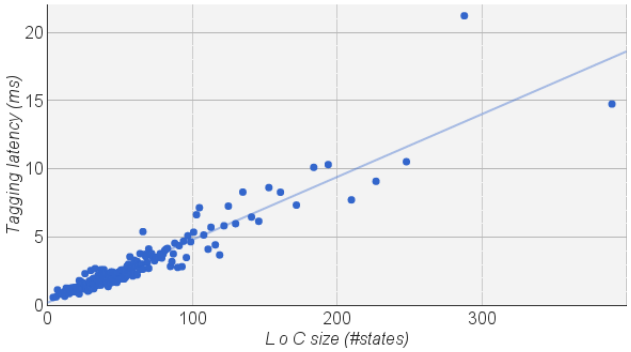

Fig. 10. Expanded lattice size vs. latency.

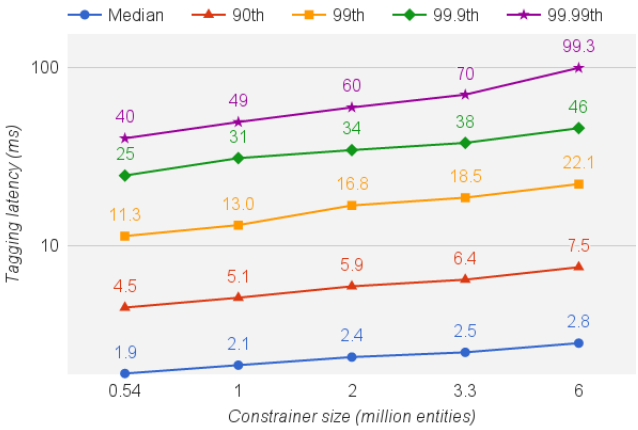

Fig. 11. Constrainer C size vs. latency.

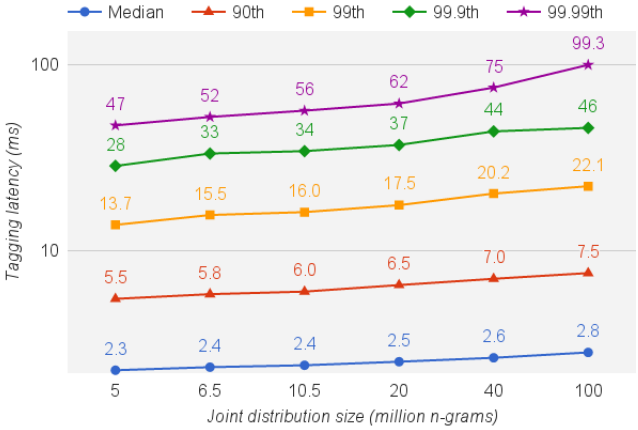

Fig. 12. Joint distribution P_T size vs. latency.

4.2. Latency measurements

Latency was measured for tagging lattices in the dataset; the benchmarking platform was an Intel Xeon E5-2690 CPU at 2.6GHz with 64GB RAM.

Figure 10 shows a linear relationship between the size (the number of states) of the expanded lattice L_T from (3) and the tagging latency. This suggests that reducing the size of L_T is key to latency reduction. We accomplished this in several ways: (a) using a pruning beam of 10 on the input lattice (paths whose total log-cost was $\geq$10 behind the best path were pruned as too improbable); (b) applying a tagging beam of 5 (taggings whose total log-cost was $\geq$5 behind the best tagging of the same path were pruned); and (c) adjusting the size of the constrainer C to achieve the desired latency-recall tradeoff.

Figure 11 shows the tradeoff between constrainer size and latency; note that both axes are in log scale. The graph shows that smaller constrainer sizes significantly reduce the search space and worst-case latencies at a relatively modest cost to recall (compare to Figure 9).

Figure 12 shows the latency reduction from using a smaller joint distribution, P_T, while keeping the constrainer size fixed at 6 million entities. The correlation with latency is modest when compressing the joint distribution, while the cost to precision and recall is somewhat significant (Figure 8). Therefore, our recommendation is to prefer a larger P_T and adjust the size of C to match the application, unless P_T is compressed using a more effective method.

5. CONCLUSIONS

In this paper, we described an FST-based semantic tagger that distills knowledge from a larger model (or from a supervised corpus) and operates directly on word lattices. The tagger's design can be useful for ASR lattice rescoring or for providing rich annotations to lattice consumers. The tagger is particularly adapted to rescoring, as it can perform "max-normalization" to preserve n-best order by normalizing probabilities by best tagging. Beam pruning of unlikely taggings keeps the lattice size manageable. Runtime is approximately linear in expanded lattice size, and we achieved a median latency of 2.1ms with F_1=.904 (lattice 1-best) compared with our baseline NED model by constraining the search space via lattice beam pruning and limiting the grammar constrainer size to the top 1 million entities. Directions for future work include finding a better way to compress the joint distribution model, parallelizing the algorithm, and using clustering to construct a better set of classes.

6. ACKNOWLEDGEMENTS

The author would like to thank Michael Riley for invaluable insights discussing the described algorithm; Michael Riley and Cyril Allauzen for their help with OpenFST and OpenGRM; and Vlad Schogol for help with infrastructure while implementing semantic data annotation.

7. REFERENCES

[1] David Nadeau and Satoshi Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[2] Jonathan G Fiscus, Jerome Ajot, John S Garofolo, and George Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proc. SIGIR*. Citeseer, 2007, vol. 7, pp. 51–57.

[3] Yushi Xu, Stephanie Seneff, Alice Li, and Joe Polifroni, "Semantic understanding by combining extended cfg parser with hmm model.," in *SLT*. Citeseer, 2010, pp. 67–72.

[4] Silviu Cucerzan, "Large-scale named entity disambiguation based on wikipedia data.," in *EMNLP-CoNLL*, 2007, vol. 7, pp. 708–716.

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 1247–1250.

[6] Lev Ratinov and Dan Roth, "Design challenges and misconceptions in named entity recognition," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2009, pp. 147–155.

[7] Alan Ritter, Sam Clark, Oren Etzioni, et al., "Named entity recognition in tweets: an experimental study," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 1524–1534.

[8] Kamel Nebhi, "Named entity disambiguation using freebase and syntactic parsing," in *Proceedings of the First International Conference on Linked Data for Information Extraction-Volume 1057*. CEUR-WS. org, 2013, pp. 50–55.

[9] Sabato Marco Siniscalchi, Jinyu Li, and Chin-Hui Lee, "A study on lattice rescoring with knowledge scores for automatic speech recognition.," in *INTERSPEECH*, 2006.

[10] Petar Aleksic, Mohammadreza Ghodsi, Assaf Michaely, Cyril Allauzen, Keith Hall, Brian Roark, David Rybach, and Pedro Moreno, "Bringing contextual information to google speech recognition," in *INTERSPEECH*. 2015, ISCA.

[11] Dilek Hakkani-Tür, Frédéric Béchet, Giuseppe Riccardi, and Gokhan Tur, "Beyond asr 1-best: Using word confusion networks in spoken language understanding," *Computer Speech & Language*, vol. 20, no. 4, pp. 495–514, 2006.

[12] Anoop Deoras, Ruhi Sarikaya, Gökhan Tür, and Dilek Z Hakkani-Tür, "Joint decoding for speech recognition and semantic tagging.," in *INTERSPEECH*, 2012, pp. 1067–1070.

[13] John Lafferty, Andrew McCallum, and Fernando Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the eighteenth international conference on machine learning, ICML*, 2001, vol. 1, pp. 282–289.

[14] Andrew Ng and Michael Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, pp. 841, 2002.

[15] Zhicheng Zheng, Xiance Si, Fangtao Li, Edward Y Chang, and Xiaoyan Zhu, "Entity disambiguation with freebase," in *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 2012, pp. 82–89.

[16] Trevor Cohn, "Efficient inference in large conditional random fields," in *European Conference on Machine Learning*. Springer, 2006, pp. 606–613.

[17] Ke Wu, Cyril Allauzen, Keith B Hall, Michael Riley, and Brian Roark, "Encoding linear models as weighted finite-state transducers.," in *INTERSPEECH*, 2014, pp. 1258–1262.

[18] 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)*. 2007, vol. 4783 of *Lecture Notes in Computer Science*, pp. 11–23, Springer, `http://www.openfst.org`.

[19] Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry 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.

[20] Christian Raymond, Frederic Bechet, Renato De Mori, and Geraldine Damnati, "On the use of finite state transducers for semantic interpretation," *Speech Communication*, vol. 48, no. 3, pp. 288–304, 2006.

[21] Christophe Servan, Christian Raymond, Frédéric Béchet, and Pascal 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.

[22] Stefan Hahn, Marco Dinarelli, Christian Raymond, Fabrice Lefevre, Patrick Lehnen, Renato De Mori, Alessandro Moschitti, Hermann Ney, and Giuseppe 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.

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

[24] Mehryar Mohri, "Weighted automata algorithms," in *Handbook of weighted automata*, pp. 213–254. Springer, 2009.

[25] Cyril Allauzen and Mehryar Mohri, "Efficient algorithms for testing the twins property," *Journal of Automata, Languages and Combinatorics*, vol. 8, no. 2, pp. 117–144, 2003.

[26] Guoguo Chen, Hainan Xu, Minhua Wu, Daniel Povey, and Sanjeev Khudanpur, "Pronunciation and silence probability modeling for asr," in *Proceedings of INTERSPEECH*, 2015.

[27] Eric Fosler, Mitch Weintraub, Steven Wegmann, Yu-Hung Kao, Sanjeev Khudanpur, Charles Galles, and Murat Saraclar, "Automatic learning of word pronunciation from data," in *the Proceedings of the International Conference on Spoken Language Processing*, 1996.

[28] Andreas Stolcke, "Entropy-based pruning of backoff language models," *arXiv preprint cs/0006025*, 2000.