LEARNING CONCEPTS THROUGH CONVERSATIONS IN SPOKEN DIALOGUE SYSTEMS

Robin Jia[*1] *Larry Heck*[2] *Dilek Hakkani-Tür*[2] *Georgi Nikolov*[2]

[1]Stanford University, Stanford, CA
[2]Google Research, Mountain View, CA
robinjia@stanford.edu larryheck,dilekh,geonik@google.com

ABSTRACT

Spoken dialogue systems must be able to recover gracefully from unexpected user inputs. In many cases, these unexpected utterances may be within the scope of the system, but include previously unseen phrases that the system cannot interpret. In this work, we augment a spoken dialogue system with the ability to learn about new concepts by conversing with the user in natural language. We present a novel model that detects phrases corresponding to such concepts, using information from a neural slotfiller as well as syntactic cues. The system then prompts the user for a definition of the detected phrases, and uses these definitions to re-parse the original utterance. We demonstrate significant gains by learning from the user, compared to a baseline system.

Index Terms— Spoken dialogue systems, interactive learning.

1. INTRODUCTION

With the rise of modern personal assistant applications, building robust spoken dialogue systems (SDSs) that can handle a wide range of possible human inputs has become an important real-world challenge. Recent work has explored the use of machine learning and deep learning techniques for various components of spoken dialogue systems [1, 2, 3, 4, 5]; such systems can handle common utterance patterns by learning from large training datasets, but may struggle when presented with unfamiliar inputs. Users may use idiosyncratic wording to refer to specific concepts; other phrases (e.g., *"my brother's house"*) have user-specific meanings. Due to the personalized nature of these utterances, it is difficult to train a generic machine learning system to handle all of these correctly.

In this work, we present a spoken dialogue system that can learn about new concepts through natural language conversation. Given a user utterance, our system identifies unexpected phrases that are likely to have semantic content. We use a multi-step process in which the system first flags individual words, then expands these words into pharses based on syntactic cues. Next, the system prompts the user to define these phrases in natural language. Finally, it uses these definitions to reinterpret the original utterance and continue the overall dialogue session. With this strategy, our system can transparently adapt to the user's particular language use, enabling users to express concepts in the way that is most natural to them.

Past work has also explored the use of system clarification questions in spoken dialogue systems. CLARIE [6] asks clarification questions when it is unable to parse the entire user utterance, but cannot query arbitrary spans of the utterance. SPIQA [7] identifies ambiguous phrases to ask about in an open-domain question answering setting, using simple heuristics and treating the question-answering component as a black box. RUDI [8] prompts the user for additional details when the user utterance is underspecified. Stoyanchev et al. [9] study the use of clarification questions to correct errors in speech recognition. Other previous work [10, 11] has explored dialog agents that can learn from users in a robot control setting.

User: *I need a reservation on Alice's birthday at Evvia.*
System: *Can you define "Alice's Birthday"?*
User: *Alice's birthday is March 9.*
System: `Reserve(restaurant=Evvia,date=3/9)`

Fig. 1. An example dialogue in our domain.

Other systems do not ask clarification questions, but can learn to understand language by interacting with users. Existing personal assitant applications (e.g., Google Now, Microsoft Cortana, Apple's Siri) allow users to choose alternative names for a limited set of entities, such as contacts. SHRDLURN [12] learns a semantic parser from scratch given user inputs and feedback about correct denotations. Szymanski and Duch [13] develop a system that can ask humans for new information about concepts in a knowledge base. KNOWBOT [14] learns scientific knowledge by asking users for natural language explanations about science test questions.

Our system is able to identify personalized concepts in user utterances, and can use user definitions to greatly improve upon a baseline system that cannot ask the user for definitions. Furthermore, our work highlights the difficulty and importance of building systems that are robust to distributional shift at test time. In most of our experiments, our system is trained solely on data that does not contain user-specific concepts, but nonetheless is expected to drive reasonable dialogues about these concepts at test time. We show that a naive baseline system that ignores the difference between training and testing performs poorly compared to a more refined approach. Finally, we release all of our data publicly, to encourage further work on this task.

2. SETUP

2.1. Scenario

We develop our approach in a goal-directed restaurant-booking scenario. Each dialogue starts with a user request for a restaurant reservation with slot-value pairs (e.g. restaurant name, location, time, date, number of people). As is often the case, the user may refer to some of the slots (location, time, and date) with personalized values. Dialogue systems typically do not know how to interpret these values a priori, because they have different meanings for different users. Examples of personalized values are shown in Table 1.

[*]Work done while the author was an intern at Google Research.

Slot	Example Personalized Values
Location	Near my workplace, near my parents' house, ...
Time	Around brunch time, after soccer practice, ...
Date	My mom's birthday, the end of the semester, ...

Table 1. A sample of personalized concepts used in our dataset.

2.2. Task

We explore the potential of learning definitions of personalized values through dialogue, as seen in Figure 1. In our task, the system has the option of asking the user to define unknown, personalized concepts in natural language. In most of our experiments, the dialogue system is trained on data that only contains impersonal concepts. Excluding personalized concepts from training has a twofold motivation. First, data without personalized concepts is straightforward to collect via crowdsourcing [15], whereas data that covers the entire spectrum of possible personalized concepts is difficult to collect, without relying on real users. Therefore, this setting approximates the resources that would be available to developers building a real spoken dialogue system. Second, we are motivated by the more general problem of building systems that are robust to previously unseen language usage patterns at test time, as such robustness is critical when interacting with real users. Therefore, we created a test-time environment that uses concepts never seen during training.

3. MODEL

3.1. Neural slotfiller

To identify slots in the user utterances, we train a recurrent neural network slotfilling model, similar to previous work [16, 17, 18]. Given an utterance $x = x_1, \ldots, x_n$, our model predicts $p_\theta(y_1, \ldots, y_n \mid x)$, where y_i is the IOB label for x_i. Our model maps each word to a pre-trained word vector from word2vec [19], then feeds these vectors through two bi-directional LSTM layers, each of which has 64 hidden units. At the top level, the model uses a softmax layer to independently predict y_i for each $i = 1, \ldots, n$. We train only on examples without personalized concepts, using mini-batch RMSProp on the loglikelihood of the gold labels and dropout of 0.5. We implemented the slotfiller in Tensorflow [20].

3.2. Interpretation

The job of the interpreter is to convert natural language phrases tagged by the slotfiller into logical values that are understood by the system. We use a simple interpration module powered by manually generated regular expressions and lexicon entries.

3.3. Identifying unknown phrases

An important part of our task is identifying *unknown phrases*, or phrases that the model can only interpret correctly by asking the user for a definition (in our dataset, these correspond to personalized phrases). Our naive baseline system assumes that the slotfiller's outputs are always correct. It computes the argmax predictions of the slotfiller, and marks a span as unknown if and only if the slotfiller tagged it as a slot, but it could not be interpreted.

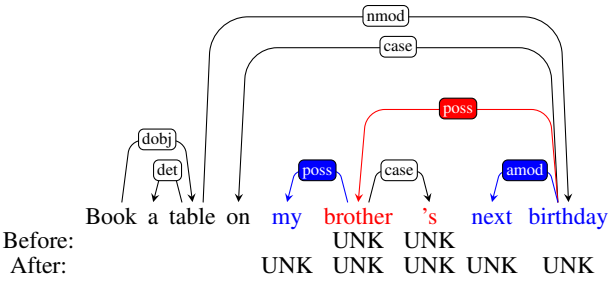

Fig. 2. Our use of the dependency tree. First, other methods mark *"brother's"* (red text) as being unknown. We follow the possessive dependency edge (red arc) upwards to the parent, then tag that node and all descendants (blue arcs and text) as unknown.

However, we expect that the slotfiller may not give reliable predictions when presented with utterances with personalized concepts, if such concepts were not seen during training. Therefore, we add some complementary strategies for identifying unknown words even in the presence of distributional shift. First, given an utterance x, we try tagging x_i as unknown if

$$\max_{\ell \in \mathcal{L}} p_\theta(y_i = \ell \mid x) < p_{\text{thresh}}, \tag{1}$$

for some threshold p_{thresh}, where $\mathcal{L}$ is the set of all IOB labels. In other words, we tag the word as unknown if the model is not very confident about its prediction. Next, from the training data $\mathcal{D}$ consisting of gold (x, y) pairs, we build a vocabulary

$$\hat{V} = \{w : \exists\, (x, y) \in \mathcal{D} \text{ s.t. } w = x_i, y_i = \texttt{O}\}. \tag{2}$$

This is the set of words that appear in the training data outside of any span (represented as `O` in IOB format). We tag x_i as unknown if the slotfiller predicts $y_i =$ `O` , but $x_i \notin \hat{V}$. This procedure is useful because the slotfiller often defaults to marking words that are unfamiliar, such as those in unknown phrases, as `O`. We also tag words that the slotfiller tagged as `I-<slot>` for some slot, but for which there was no preceding `B-<slot>`. Finally, we tag as unknown any words that the baseline tagged as unknown.

The above procedure has relatively high precision, but may miss many words, resulting in unnatural phrases that are short and fragmented. To remedy this problem, we use the dependency parse of the sentence to expand the set of words we tag, as seen in Figure 2. For each tagged word, we first traverse the dependency tree upwards along edges corresponding to compound nouns, adjectival modifiers, or posessive modifiers—this procedure helps us locate the root of the noun phrase that this word belongs to. Then, we mark this node and all its descendents as being unknown. This procedure guides the system to query syntactically coherent phrases.

Finally, we join adjacent words tagged as unknown into contiguous phrases. The system then asks the user to define each phrase.

3.4. Incorporating user-supplied definitions

After the system asks the user to define each unknown phrase, it must then use the user-supplied definitions to reinterpret the original first-turn utterance. We found a simple strategy to work surprisingly well: we directly replace the unknown phrase from the first-turn utterance with the entire user-supplied definiton. This procedure may

Test set	Precision	Recall	F1
Basic	79.0	81.4	80.2
Personalized	49.0	48.7	48.9

Table 2. Slotfiller phrase-level metrics on the two test sets.

result in an ungrammatical sentence, but we find that the slotfiller is able to ignore unnecessary words and perform quite well. Then, we parse this new sentence with the neural slotfiller and extract slot-value pairs using the interpretation module.

4. EXPERIMENTS

4.1. Data Collection

We collected data using crowdsourcing, in a manner similar to [15]. To collect first-turn data, we start by randomly generating a set of "logical forms," which for us are sets of slot-value pairs. To generate a logical form, we randomly choose one to three slots from our schema, then randomly sample a value for each slot from a predefined list of possible values. We send each logical form to two crowd-workers, and ask each to formulate three different natural language requests using the given slot-value pairs. Other crowd-workers then label these utterances by highlighting the parts of the sentence corresponding to each slot; this step also serves as a quality control filter.

Next, we collect the second-turn data. We synthetically generate system-turn questions by extracting spans in the collected utterances corresponding to user-specific values. For each such span, we generate the question *"Can you define '<phrase>'?"* We then launch additional tasks for crowd-workers, in which they are given the first-turn utterance, the system question, and a non-personalized value that they are told to use in their response to the system question. For example, if the user-specific concept was *"Alice's birthday"*, we might tell the crowd workers to use the value `Mar 9`. Again, each task is sent to two crowd-workers, who generate three different natural language responses each. We run labeling on these utterances as well.

Our final dataset contains 2193 first-turn "basic" utterances without personalized concepts, and 594 first-turn utterances with personalized concepts. We split the basic utterances into 1534 train, 219 development, and 440 test examples; we split the personalized first-turn utterances into 178 development and 416 test examples. For the personalized data, each first-turn utterance has multiple accompanying follow-up turns. Taking these into account, we had 1281 distinct development conversations, and 3014 distinct test conversations. We have made all of our data publicly available [1].

4.2. Slotfiller performance

First, we evaluate the neural slotfiller on its own, on both basic and personalized data, as shown in Table 2. On the basic test set, the slotfiller achieves an overall phrase-level F1 score of 80.2. Manual error analysis reveals that many errors are due to inconsistencies in human labeling (e.g., whether *"at 7pm"* is labeled as a time, or only *"7pm"*). Therefore, we are confident that, when presented with utterances without personalized expressions, our model is quite good at identifying slots. However, on the personalized test set, the slotfiller performs much worse, with an F1 score of 48.9. This result confirms that the slotfiller cannot generalize well to this out-of-domain data, despite using pre-trained word embeddings.

[1] Our data is available for download at `http://stanford.edu/~robinjia/data/concept_learning_data.tar.gz`

System	Word-level	Phrase-level
Naive baseline	73.4/46.4/56.8	28.7/27.3/28.0
Scores only	73.3/60.2/66.1	29.6/32.4/31.0
OOV only	**77.2**/72.9/75.0	30.1/39.1/34.0
Scores + OOV	75.5/74.7/75.1	30.5/**40.0**/34.6
Scores + syntax	69.9/77.2/73.3	**41.1**/39.6/**40.3**
OOV + syntax	71.5/87.1/**78.6**	35.1/38.9/36.9
Scores + OOV + syntax	70.0/**88.5**/78.1	36.1/**40.0**/37.9

Table 3. Evaluation of unknown phrase identification. For each of the evaluation modes, we report precision/recall/F1.

System	Precision	Recall	F1
No questions baseline	96.1	39.6	56.1
Naive baseline	95.1	49.5	65.1
Our model	95.3	56.9	71.3
Gold questions oracle	93.9	76.7	84.4

Table 4. Slot-level results using simulated end-to-end evaluation on the personalized test set.

4.3. Identifying unknown phrases

Next, we evaluate our system's ability to extract unknown phrases from user utterances, as shown in Table 3. We measure both word-level and phrase-level F1 score. The naive baseline has very low recall, particularly at the word level, since the slotfiller often ignores unknown phrases by tagging them as `O`.

Next, we evaluate our more refined model, using slotfiller confidence scores, vocabulary information, and syntactic information. In all experiments, we used $p_{\text{thresh}} = 0.7$. All versions of our model outperform the baseline system. As expected, removing syntactic features greatly increases word-level recall, but at the cost of some precision. Adding the slotfiller's confidence scores do not help much compared to using out-of-vocabulary information alone. Therefore, in all following experiments, we only use out-of-vocabulary and syntactic information.

4.4. End-to-end evaluation

To measure our overall system performance, we performed a simulated end-to-end evaluation. We avoid conducting direct end-to-end

System	Basic	Personalized
Naive baseline	0.30	1.03
Our model	0.38	1.20
Gold questions oracle	0.00	1.08

Table 5. Number of questions asked per dialogue by the system, compared to the gold questions, on both the basic and personalized test sets.

Method	Model questions	Gold questions
Full substitution	95.3/56.9/**71.3**	93.9/76.7/**84.4**
Slotfiller-based	96.0/54.4/69.4	95.3/71.2/81.5
Gold span oracle	95.5/56.6/71.1	93.7/74.2/82.8

Table 6. Simulated end-to-end evaluation using different strategies for using the user-supplied definitions. We report precision/recall/F1, when using either the model's predicted questions or the gold questions.

evaluation with human users for two reasons. First, human evaluation is time-consuming and expensive, which hampers development. Second, since we tell our crowdworkers what value to use in their response, they may give a helpful answer even when the system question is nonsensical, which would cause us to overestimate how well our system works.

Instead, we simulate whole dialogues using our existing data, and evaluate based on interpreted slot-level F1 score. For each question that our system generates, we decide if this question is "similar enough" to a gold question. If it is, we provide the system the human response to the gold question; otherwise, we assume the user provides no response to this question. We define a predicted span to be "similar enough" to a gold span if the spans overlap and are identical except for stopwords and punctuation. We use a predefined list of stopwords from NLTK [21], combined with a few additional words (e.g. *"restaurant"*) that are not semantically important in our particular domain. This is a strict definition of similarity: in practice, a user may give a helpful definition even if the system drops or adds some content words.

We compare our best system and the naive baseline to an even simpler baseline system that does not ask any questions, and only looks at the first-turn utterance. We also compare to an oracle system that always asks the gold questions. In Table 4, we measure both slot-level F1, after the finishing the dialogue and running interpretation. In Table 5, we count the number of questions asked by the system—all else being equal, shorter dialogues are more convenient for the user. Note that the naive model asks 0.30 questions per dialogue on the basic data. These a consequence of blind spots in our simple interpretation module, and are not indicative of our unknown phrase detection routine reporting false positives.

Next, we evaluate our strategy of substituting the entire user-supplied definition for the unknown phrase. We compare to an alternative in which we run the slotfiller on the definition, extract exactly one slot, and replace the unknown phrase in the first turn with the extracted phrase. We also compare to an oracle that replaces the unknown phrase with the gold span tagged in the definition by human raters. In Table 6, we see that our strategy is very effective, doing better than the slotfiller-based approach, and matching the oracle performance. We believe that preserving the context words in both the definition and the first-turn utterance helps the slotfiller.

4.5. Adding personalized data

Finally, we measure how much the system can improve when given a small amount of training data involving personalized concepts—in our case, the 178-example personalized development set. This scenario simulates what might happen after an initial system is deployed, and user data with personalized concepts is now available.

System	Precision	Recall	F1	Num. Q's
Fine-tuned slotfiller				
Naive	94.4	66.9	78.3	1.26
Our model	94.9	56.8	71.1	1.20
Lexicon				
Naive	96.5	51.3	67.0	0.66
Our model	94.9	57.9	71.9	0.66
Fine-tuned + lexicon				
Naive	94.7	67.9	79.1	0.36
Our model	94.5	58.4	72.2	0.49

Table 7. End-to-end slot-level metrics and number of questions asked on the personalized test set, with personalized training data.

We explore two different ways to use this additional data. First, we retrain the neural slotfiller on this new personalized data in addition to the original basic training data, using the exact same hyperparameters. Second, we build a lexicon mapping gold unknown phrases in the personalized development data to user-supplied definitions. During evaluation, when the system identifies an unknown phrase, it first checks the lexicon: if the exact phrase is there, it uses the definition from the lexicon instead of asking the user. This second method assumes that the new personalized data comes from the same user that the system is currently interacting with.

Table 7 shows the effects of these adjustments, both independently and together. After fine-tuning the slotfiller, the naive baseline outperforms our full system; now that the slotfiller has seen data on personalized concepts, trusting its predictions on similar data at test time is a good idea. As expected, using the lexicon reduces the number of questions asked. Slot-level performance also increases slightly, because the lexicon helps the system when it asks a question that is slightly off from the gold question.

5. DISCUSSION

Much future work remains to improve our system, beginning with the problem of natural language generation (NLG). We used a simple NLG module, with just a single template. If we could predict the type of unknown phrases, we might be able to use a richer set of type-specific templates (e.g. asking *"When is Bob's birthday?"* instead of simply *"Can you define 'Bob's birthday'?"*). It would also be interesting to learn to handle cases where the phrase in the utterance should not be directly copied into the question, for example in the case of personal pronouns (e.g., if the user says *"my birthday"*, the system should ask, *"When is your birthday?"*). Recent work on neural NLG [2] could be useful to learn these non-trivial patterns.

Our experiments showed that different techniques work best depending on whether a given utterance is similar enough to what was seen during training time. A natural next step would be to identify distributional shift on the fly, so that the system can trust its machine learning components on in-domain data, but fall back on more robust strategies on data it was not trained to handle. Ultimately, any successful spoken dialogue system must be able to adapt to new users and new language use at test time.

Acknowledgments. We would like to thank Ankur Bapna for his help with the neural slotfiller. Robin is supported in part by NSF Graduate Research Fellowship Program Grant No. DGE-114747.

6. REFERENCES

[1] M. Henderson, B. Thomson, and S. J. Young, "Deep Neural Network Approach for the Dialog State Tracking Challenge," in *Proceedings of SIGdial*, 2013.

[2] Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, September 2015, pp. 1711–1721, Association for Computational Linguistics.

[3] Emmanuel Ferreira and Fabrice Lefèvre, "Reinforcement-learning based dialogue system for human-robot interactions with socially-inspired rewards," *Comput. Speech Lang.*, vol. 34, no. 1, pp. 256–274, Nov. 2015.

[4] David Griol, Lluís F. Hurtado, Encarna Segarra, and Emilio Sanchis, "A statistical approach to spoken dialog systems design and evaluation," *Speech Commun.*, vol. 50, no. 8-9, pp. 666–682, Aug. 2008.

[5] Gokhan Tur and Renato De Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, John Wiley and Sons, 2011.

[6] Matthew Purver, "CLARIE: Handling clarification requests in a dialogue system," *Research on Language and Computation*, vol. 4, no. 2-3, pp. 259–288, Oct. 2006.

[7] Chiori Hori, Takaaki Hori, Hideki Isozaki, Eisaku Maeda, Shigeru Katagiri, and Sadaoki Furui, "Deriving disambiguous queries in a spoken interactive odqa system," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[8] David Schlangen, Alex Lascarides, and Ann Copestake, "Resolving underspecification using discourse information," in *Proceedings of SemDial-5 (BiDialog)*. 2001.

[9] Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg, "Modelling human clarification strategies," in *Proceedings of SIGdial*, 2013.

[10] Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso, "Learning environmental knowledge from task-based human-robot dialog," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[11] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone, "Learning to interpret natural language commands through human-robot dialog," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2013.

[12] Sida I. Wang, Percy Liang, and Christopher D. Manning, "Learning language games through interaction," in *Association for Computational Linguistics (ACL)*, 2016.

[13] Julian Szymanski and Wlodzislaw Duch, "Semantic memory knowledge acquisition through active dialogues," in *Proceedings of International Joint Conference on Neural Networks*, 2007.

[14] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi, "Learning knowledge graphs for question answering through conversational dialog," in *North American Association for Computational Linguistics (NAACL)*, 2015.

[15] Yushi Wang, Jonathan Berant, and Percy Liang, "Building a semantic parser overnight," in *Association for Computational Linguistics (ACL)*, 2015.

[16] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tür, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig, "Using recurrent neural networks for slot filling in spoken language understanding," *Trans. Audio, Speech and Lang. Proc.*, vol. 23, no. 3, pp. 530–539, Mar. 2015.

[17] Aaron Jaech, Larry Heck, and Mari Ostendorf, "Domain adaptation of recurrent neural networks for natural language understanding," in *Interspeech*, 2016.

[18] Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang, "Multi-domain joint semantic frame parsing using bi-directional rnn-lstm," in *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*. June 2016, ISCA.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., pp. 3111–3119. Curran Associates, Inc., 2013.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[21] Steven Bird, Edward Loper, and Ewan Klein, *Natural Language Processing with Python*, O'Reilly Media Inc., 2009.