# Fast and Scalable Expansion of Natural Language Understanding Functionality for Intelligent Agents

**Anuj Goyal, Angeliki Metallinou, Spyros Matsoukas**
Amazon Alexa Machine Learning
{anujgoya, ametalli, matsouka}@amazon.com

## Abstract

Fast expansion of natural language functionality of intelligent virtual agents is critical for achieving engaging and informative interactions. However, developing accurate models for new natural language domains is a time and data intensive process. We propose efficient deep neural network architectures that maximally re-use available resources through transfer learning. Our methods are applied for expanding the understanding capabilities of a popular commercial agent and are evaluated on hundreds of new domains, designed by internal or external developers. We demonstrate that our proposed methods significantly increase accuracy in low resource settings and enable rapid development of accurate models with less data.

## 1 Introduction

Voice powered artificial agents have become widespread among consumer devices, with agents like Amazon Alexa, Google Now and Apple Siri being popular and widely used. Their success relies not only on accurately recognizing user requests, but also on continuously expanding the range of requests that they can understand. An ever growing set of functionalities is critical for creating an agent that is engaging, useful and human-like.

This presents significant scalability challenges regarding rapidly developing the models at the heart of the natural language understanding (NLU) engines of such agents. Building accurate models for new functionality typically requires collection and manual annotation of new data resources, an expensive and lengthy process, often requiring highly skilled teams. In addition, data collected from real user interactions is very valuable for developing accurate models but without an accurate model already in place, the agent will not enjoy widespread use thereby hindering collection of high quality data.

Presented with this challenge, our goal is to speed up the natural language expansion process for Amazon Alexa, a popular commercial artificial agent, through methods that maximize re-usability of resources across areas of functionality. Each area of Alexa's functionality, e.g., Music, Calendar, is called a *domain*. Our focus is to a) increase accuracy of low resource domains b) rapidly build new domains such that the functionality can be made available to Alexa's users as soon as possible, and thus start benefiting from user interaction data. To achieve this, we adapt recent ideas at the intersection of deep learning and transfer learning that enable us to leverage available user interaction data from other areas of functionality.

To summarize our contributions, we describe data efficient deep learning architectures for NLU that facilitate knowledge transfer from similar tasks. We evaluate our methods at a much larger scale than related transfer learning work in NLU, for fast and scalable expansion of hundreds of new natural language domains of Amazon Alexa, a commercial artificial agent. We show that our methods achieve significant performance gains in low resource settings and enable building accurate functionality faster during early stages of model development by reducing reliance on large annotated datasets.

## 2 Related Work

Deep learning models, including Long-Short term memory networks (LSTM) (Gers et al., 1999), are state of the art for many natural language processing tasks (NLP), such as sequence labeling (Chung et al., 2014), named entity recognition (NER) (Chiu and Nichols, 2015) and part of speech (POS) tagging (Huang et al., 2015).

Multitask learning is also widely applied in NLP, where a network is jointly trained for multiple related tasks. Multitask architectures have been succesfully applied for joint learning of NER, POS, chunking and supertagging tasks, as in (Collobert et al., 2011; Collobert and Weston, 2008; Søgaard and Goldberg, 2016).

Similarly, transfer learning addresses the transfer of knowledge from data-rich source tasks to under-resourced target tasks. Neural transfer learning has been successfully applied in computer vision tasks where lower layers of a network learn generic features that are transferred well to different tasks (Zeiler and Fergus, 2014; Krizhevsky et al., 2012). Such methods led to impressive results for image classification and object detection (Donahue et al., 2014; Sharif Razavian et al., 2014; Girshick et al., 2014) In NLP, transferring neural features across tasks with disparate label spaces is relatively less common. In (Mou et al., 2016), authors conclude that network transferability depends on the semantic relatedness of the source and target tasks. In cross-language transfer learning, (Buys and Botha, 2016) use weak supervision to project morphology tags to a common label set, while (Kim et al., 2017a) transfer lower layer representations across languages for POS tagging. Other related work addresses transfer learning where source and target share the same label space, while feature and label distributions differ, including deep learning methods (Glorot et al., 2011; Kim et al., 2017b), and earlier domain adaptation methods such as EasyAdapt (Daumé III, 2007), instance weighting (Jiang and Zhai, 2007) and structural correspondence learning (Blitzer et al., 2006).

Fast functionality expansion is critical in industry settings. Related work has focused on scalability and ability to learn from few resources when developing a new domain, and includes zero-shot learning (Chen et al., 2016; Ferreira et al., 2015), domain attention (Kim et al., 2017c), and scalable, modular classifiers (Li et al., 2014). There is a multitude of commercial tools for developers to build their own custom natural language applications, including Amazon Alexa ASK (Kumar et al., 2017), DialogFlow by Google (DialogFlow) and LUIS by Microsoft (LUIS). Along these lines, we propose scalable methods that can be applied for rapid development of hundreds of low resource domains across disparate label spaces.

## 3 NLU Functionality Expansion

We focus on Amazon Alexa, an intelligent conversational agent that interacts with the user through voice commands and is able to process requests on a range of natural language domains, e.g., playing music, asking for weather information and editing a calendar. In addition to this *built-in* functionality that is designed and built by internal developers, the Alexa Skills Kit (ASK) (Kumar et al., 2017) enables external developers to build their own *custom* functionality which they can share with other users, effectively allowing for unlimited new capabilities. Below, we describe the development process and challenges associated with natural language domain expansion.

For each new domain, the internal or external developers define a set of *intents* and *slots* for the target functionality. Intents correspond to user intention, e.g., 'FindRecipeIntent', and slots correspond to domain-specific entities of interest e.g.,'FoodItem'. Developers also define a set of commonly used utterances that cover the core use cases of the functionality, e.g., 'find a recipe for chicken'. We call those *core utterances*. In addition, developers need to create *gazetteers* for their domain, which are lists of slot values. For example, a gazetteer for 'FoodItem' will contain different food names like 'chicken'. We have developed infrastructure to allow internal and external teams to define their domain, and create or expand linguistic resources such as core utterances and gazetteers. We have also built tools that enable extracting *carrier phrases* from the example utterances by abstracting the utterance slot values, such as 'find a recipe for {FoodItem}'. The collection of carrier phrases and gazetteers for a domain is called a *grammar*. Grammars can be sampled to generate synthetic data for model training. For example, we can generate the utterance 'find a recipe for pasta' if the latter dish is contained in the 'FoodItem' gazetteer.

Next, developers enrich the linguistic resources available for a new domain, to cover more linguistic variations for intents and slots. This includes creating bootstrap data for model development, including collecting utterances that cover the new functionality, manually writing variations of example utterances, and expanding the gazetteer values. In general, this is a time and data intensive process. External developers can also continuously enrich the data they provide for their custom domain. However, external developers typically lack the time, resources or expertise to provide rich datasets, therefore in practice custom domains are significantly under-resourced compared to built-in domains.

Once the new domain model is bootstrapped using the collected datasets, it becomes part of Alexa's natural language functionality and is available for user interactions. The data from such user interactions can be sampled and annotated in order to provide additional targeted training data for improving the accuracy of the domain. A good bootstrap model accuracy will lead to higher user engagement with the new functionality and hence to a larger opportunity to learn from user interaction data.

Considering these challenges, our goal is to reduce our reliance on large annotated datasets for a new domain by re-using resources from existing domains. Specifically, we aim to achieve higher model accuracy in low resource settings and accelerate new domain development by building good quality bootstrap models faster.

## 4 Methodology

In this section, we describe transfer learning methods for efficient data re-use. Transfer learning refers to transferring the knowledge gained while performing a task in a *source* domain $D_s$ to benefit a related task in a *target* domain $D_t$. Typically, we have a large dataset for $D_s$, while $D_t$ is an under-resourced new task. Here, the target domain is the new built-in or custom domain, while the source domain contains functionality that we have released, for which we have large amounts of data. The tasks of interest in both $D_s$ and $D_t$ are the same, namely slot tagging and intent classification. However $D_s$ and $D_t$ have different label spaces $Y_s$ and $Y_t$, because a new domain will contain new intent and slot labels compared to previously released domains.

### 4.1 DNN-based natural language engine

We first present our NLU system where we perform slot tagging (ST) and intent classification (IC) for a given input user utterance. We are inspired by the neural architecture of (Søgaard and Goldberg, 2016), where a multi-task learning architecture is used with deep bi-directional Recurrent Neural Networks (RNNs). Supervision for the different tasks happens at different layers. Our neural network contains three layers

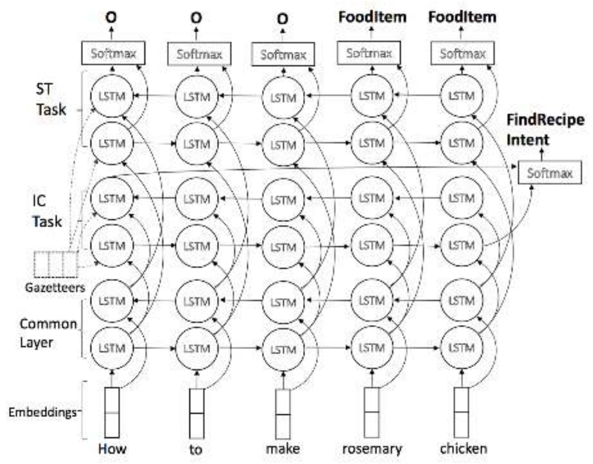

Figure 1: Multitask stacked bi-LSTM architecture for ST and IC, with a shared bottom layer, two separate top layers for ST and IC. Gazetteer features can be added as optional input to the ST and IC layers during the fine-tuning stage. (see also Sec. 4.2)

of bi-directional Long Short Term Memory networks (LSTMs) (Graves and Schmidhuber, 2005; Hochreiter and Schmidhuber, 1997). The two top layers are optimized separately for the ST and IC tasks, while the common bottom layer is optimized for both tasks, as shown in Figure 1.

Specifically let $r_t^c$ denote the common representation computed by the bottommost bi-LSTM for each word input at time $t$. The ST forward LSTM layer learns a representation $r_t^{ST,f} = \phi(r_t^c, r_{t-1}^{ST})$, where $\phi$ denotes the LSTM operation. The IC forward LSTM layer learns $r_t^{IC,f} = \phi(r_t^c, r_{t-1}^{IC})$. Similarly, the backward LSTM layers learn $r_t^{ST,b}$ and $r_t^{IC,b}$. To obtain the slot tagging decision, we feed the ST bi-LSTM layer's output per step into a softmax, and produce a slot label at each time step (e.g., at each input word). For the intent decision, we concatenate the last time step from the forward LSTM with the first step of the backward LSTM, and feed it into a softmax for classification:

$$r_t^{slot} = r_t^{ST,f} \oplus r_t^{ST,b}, r^{intent} = r_T^{IC,f} \oplus r_1^{IC,b}$$
$$\hat{S}_t = softmax(W_s r_t^{slot} + b_s)$$
$$\hat{I} = softmax(W_I r^{intent} + b_I)$$

where $\oplus$ denotes concatenation. $W_s, W_I, b_s, b_I$ are the weights and biases for the slot and intent softmax layers respectively. $\hat{S}_t$ is the predicted slot tag per time step (per input word), and $\hat{I}$ is the predicted intent label for the sentence.

The overall objective function for the multi-task network combines the IC and ST objectives.

Therefore we jointly learn a shared representation $r_t^c$ that leverages the correlations between the related IC and ST tasks, and shares beneficial knowledge across tasks. Empirically, we have observed that this multitask architecture achieves better results than separately training intent and slot models, with the added advantage of having a single model, and a smaller total parameter size.

In our setup, each input word is embedded into a 300-dimensional embedding, where the embeddings are estimated from our data. We also use pre-trained word embeddings as a separate input, that allows incorporating unsupervised word information from much larger corpora (FastText (Bojanowski et al., 2016)). We encode slot spans using the IOB tagging scheme (Ramshaw and Marcus, 1995). When we have available *gazetteers* relevant to the ST task, we use gazetteer features as an additional input. Such features are binary indicators of the presence of an n-gram in a gazetteer, and are common for ST tasks (Radford et al., 2015; Nadeau and Sekine, 2007).

## 4.2 Transfer learning for the DNN engine

Typically, a new domain $D_t$ contains little available data for training the multitask DNN architecture of Sec 4.1. We propose to leverage existing data from mature released domains (source $D_s$) to build generic models, which are then adapted to the new tasks (target $D_t$).

We train our DNN engine using labeled data from $D_s$ in a supervised way. The source slot tags space $Y_s^{slot}$ and intent label space $Y_s^{intent}$ contain labels from previously released slots and intents respectively. We refer to this stage as *pre-training*, where the stacked layers in the network learn to generate features which are useful for the ST and IC tasks of $D_s$. Our hypothesis is that such features will also be useful for $D_t$. After pre-training is complete, we replace the top-most affine transform and softmax layers for IC and ST with layer dimensions that correspond to the target label space for intents and slots respectively, i.e., $Y_t^{intent}$ and $Y_t^{slot}$. The network is then trained again using the available target labeled data for IC and ST. We refer to this stage as *fine-tuning* of the DNN parameters for adapting to $D_t$.

A network can be pre-trained on large datasets from $D_s$ and later fine tuned separately for many low resource new domains $D_t$. In some cases, when developing a new domain $D_t$, new domain-specific information becomes available, such as domain gazetteers (which were not available at pre-training). To incorporate this information during fine-tuning, we add gazetteer features as an extra input to the two top-most ST and IC layers, as shown in Figure 1. We found that adding new features during fine-tuning significantly changes the upper layer distributions. Therefore, in such cases, it is better to train the ST and IC layers from scratch and only transfer and fine-tune weights from the common representation $r_c$ of the bottom layer. However, when no gazetteers are available, it is beneficial to pre-train all stacked Bi-LSTM layers (common, IC and ST), except from the task-specific affine transform leading to the softmax.

## 4.3 Baseline natural language engine

While DNNs are strong models for both ST and IC, they typically need large amounts of training data. As we focus on under-resourced functionality, we examine an alternative baseline that relies on simpler models; namely a Maximum Entropy (MaxEnt) (Berger et al., 1996) model for intent classification and a Conditional Random Field (CRF) (Lafferty et al., 2001) model for slot tagging. MaxEnt models are regularized log-linear models that have been shown to be effective for text classification tasks (Berger et al., 1996). Similarly, CRFs have been popular tagging models in the NLP literature (Nadeau and Sekine, 2007) prior to the recent growth in deep learning. In our experience, these models require less data to train well and represent strong baselines for low resource classification and tagging tasks.

## 5 Experiments and Results

We evaluate the transfer learning methods of Section 4.2 for both custom and built-in domains, and compare with baselines that do not benefit from knowledge transfer (Sections 4.1, 4.3). We experiment with around 200 developer defined custom domains, whose statistics are presented in Table 1. Looking at the median numbers, which are less influenced by a few large custom domains compared to mean values, we note that typically developers provide just a few tens of example phrases and few tens of values per gazetteer (slot gazetteer size). Therefore, most custom domains are significantly under-resourced. We also select three new built-in domains, and evaluate them at various early stages of domain development. Here, we assume that variable amounts of training data gradually become available, including bootstrap and user interaction data.

We pre-train DNN models using millions of annotated utterances from existing mature built-in domains. Each annotated utterance has an associated domain label, which we use to make sure that the pre-training data does not contain utterances labeled as any of the custom or built-in target domains. After excluding the target domains, the pre-training data is randomly selected from a variety of mature Alexa domains covering hundreds of intents and slots across a wide range of natural language functionality. For all experiments, we use L1 and L2 to regularize our DNN, CRF and MaxEnt models, while DNNs are additionally regularized with dropout.

The test sets contain user data, annotated for each custom or built-in domain. For custom domains, test set size is a few hundred utterances per domain, while for built-in domains it is a few thousand utterances per domain. Our metrics include standard F1 scores for the SC and IC tasks, and a sentence error rate (SER) defined as the ratio of utterances with at least one IC or ST error over all utterances. The latter metric combines IC and ST errors per utterance and reflects how many utterances we could not understand correctly.

| Data type | Mean | Median |
|---|---|---|
| number of intents | 8.02 | 3 |
| number of slots | 2.07 | 1 |
| slot gazetteer size | 441.35 | 11 |
| number of example phrases | 268.11 | 42 |

Table 1: Statistics of data for around 200 developer defined custom domains

## 5.1 Results for custom developer domains

For the custom domain experiments, we focus on a low resource experimental setup, where we assume that our only target training data is the data provided by the external developer. We report results for around 200 custom domains, which is a subset of all domains we support. We compare the proposed transfer learning method, denoted as *DNN Pretrained*, with the two baseline methods described in sections 4.1 and 4.3, denoted as *DNN Baseline* and *CRF/MaxEnt Baseline*, respectively. For training the baselines, we use the available data provided by the developer for each domain, e.g., example phrases and gazetteers. From these resources, we create grammars and we sample them to generate 50K training utterances per domain, using the process described in Section 3. This training data size was selected empirically based on baseline model accuracy. The generated utterances may contain repetitions for domains where the external developer provided a small amount of example phrases and few slot values per gazetteer. For the proposed method, we pre-train a DNN model on 4 million utterances and fine tune it per domain using the 50K grammar utterances of that domain and any available gazetteer information (for extracting gazetteer features). In Table 2, we show the mean and median across custom domains for $F1_{slot}$, $F1_{intent}$ and $SER$.

Table 2 shows that the CRF and MaxEnt models present a strong baseline and generally outperform the DNN model without pretraining, which has a larger number of parameters. This suggests that the baseline DNN models (without pretraining) cannot be trained robustly without large available training data. The proposed pre-trained DNN significantly outperforms both baselines across all metrics (paired t-test, $p < .01$). Median $SER$ reduces by around 14% relative when we use transfer learning compared to both baselines. We are able to harness the knowledge obtained from data of multiple mature source domains $D_s$ and transfer it to our under-resourced target domains $D_t$, across disparate label spaces.

To investigate the effect of semantic similarity across source and target domains we selected a subset of 30 custom domains with high semantic similarity with the source tasks. Semantic similarity was computed by comparing the sentence representations computed by the common bi-LSTM layer across source and target sentences, and selecting target custom domains with sentences close to at least one of the source tasks. For these 30 domains, we observed higher gains of around 19% relative median SER reduction. This corroborates observations of (Mou et al., 2016), that neural feature transferability for NLP depends on the semantic similarity between source and target. In our low resource tasks, we see a benefit from transfer learning and this benefit increases as we select more semantically similar data.

Our approach is scalable and is does not rely on manual domain-specific annotations, besides developer provided data. Also, pretrained DNN models are about five times faster to train during the fine-tuning stage, compared to training the model from scratch for each custom domain,

| Approach | $F1_{Intent}$ Mean | $F1_{Intent}$ Median | $F1_{Slot}$ Mean | $F1_{Slot}$ Median | $SER$ Mean | $SER$ Median |
|---|---|---|---|---|---|---|
| Baseline CRF/MaxEnt | 94.6 | 96.6 | 80.0 | 91.5 | 14.5 | 9.2 |
| Baseline DNN | 91.9 | 95.9 | 85.1 | 92.9 | 14.7 | 9.2 |
| Proposed Pretrained DNN * | **95.2** | **97.2** | **88.6** | **93.0** | **13.1** | **7.9** |

Table 2: Results for around 200 custom developer domains. For F1, higher values are better, while for SER lower values are better. * denotes statistically significant SER difference compared to both baselines.

which speeds up model turn-around time.

## 5.2 Results for built-in domains

We evaluate our methods on three new built-in domains referred here as domain A (5 intents, 36 slot types), domain B (2 intents, 17 slot types) and domain C (22 intents, 43 slot types). Table 3 shows results for domains A, B and C across experimental early stages of domain development, where different data types and amounts of data per data type gradually become available. Core data refers to core example utterances, bootstrap data refers to domain data collection and generation of synthetic (grammar) utterances, and user data refers to user interactions with our agent. As described in Section 3, the collection and annotation of these data sources is a lengthy process. Here we evaluate whether we can accelerate the development process by achieving accuracy gains in early, low resource stages, and bootstrap a model faster.

For each data setting and size, we compare our proposed pretrained DNN models with the baseline CRF/MaxEnt baseline, which is the better performing baseline of Section 5.1. Results for the non pre-trained DNN baseline are similar, and omitted for lack of space. Our proposed DNN models are pre-trained on 4 million data from mature domains and then fine tuned on the available target data. The baseline CRF/MaxEnt models are trained on the available target data. Note that the datasets of Table 3 represent early stages of model development and do not reflect final training size or model performance. The types of target data slightly differ across domains according to domain development characteristics. For example, for domain B there was very small amount of core data available and it was combined with the bootstrap data for experiments.

Overall, we notice that our proposed DNN pretraining method improves performance over the CRF/MaxEnt baseline, for almost all data settings. As we would expect, we see the largest gains for the most low resource data settings. For example, for domain A, we observe a 7% and 5% relative

| Train Set | Size | Method | $F1_{intent}$ | $F1_{slot}$ | $SER$ |
|---|---|---|---|---|---|
| Domain A (5 intents, 36 slots) | | | | | |
| Core* data | 500 | Baseline | 85.0 | 63.9 | 51.9 |
| | | Proposed | 86.6 | 66.6 | 48.2 |
| Bootstrap data* | 18K | Baseline | 86.1 | 72.8 | 49.6 |
| | | Proposed | 86.9 | 73.8 | 47.0 |
| Core + user data* | 3.5K | Baseline | 90.4 | 74.3 | 40.5 |
| | | Proposed | 90.1 | 75.8 | 37.9 |
| Core + bootstrap + user data | 43K | Baseline | 92.1 | 80.6 | 33.4 |
| | | Proposed | 91.9 | 80.8 | 32.8 |
| Domain B (2 intents, 17 slots) | | | | | |
| Bootstrap data* | 2K | Baseline | 97.0 | 94.7 | 10.1 |
| | | Proposed | 97.8 | 95.3 | 6.3 |
| User data | 2.5K | Baseline | 97.0 | 94.7 | 8.2 |
| | | Proposed | 97.1 | 96.4 | 7.1 |
| Bootstrap + user data* | 52K | Baseline | 96.7 | 95.2 | 8.2 |
| | | Proposed | 97.0 | 96.6 | 6.4 |
| Domain C (22 intents, 43 slots) | | | | | |
| Core* data | 300 | Baseline | 77.9 | 47.8 | 64.2 |
| | | Proposed | 85.6 | 46.6 | 51.8 |
| Bootstrap data* | 26K | Baseline | 46.1 | 65.8 | 64.0 |
| | | Proposed | 49.1 | 68.9 | 62.8 |
| Core + bootstrap. + user data* | 126K | Baseline | 92.3 | 78.3 | 28.1 |
| | | Proposed | 92.7 | 72.7 | 31.9 |

Table 3: Results on domains A, B and C for the proposed pretrained DNN method and the baseline CRF/MaxEnt method during experimental early stages of domain development. * denotes statistically significant SER difference between proposed and baseline

SER improvement on core and bootstrap data settings respectively. The performance gain we obtain on those early stages of development brings us closer to our goal of rapidly bootstrapping models with less data. From domains A and C, we also notice that we achieve the highest performance in settings that leverage user data, which highlights the importance of such data. Note that the drop in $F_{intent}$ for domain C between core and bootstrap data is because the available bootstrap data did not contain data for all of the 22 intents of domain C. Finally, we notice that the gain from transfer learning diminishes in some larger data settings, and we may see degradation (domain C, 126K data setting). We hypothesize that as larger training data becomes available it may be better to not pre-train or pre-train with source data that are semantically similar to the target. We will investigate this as part of future work.

## 6 Conclusions and Future Work

We have described the process and challenges associated with large scale natural language functionality expansion for built-in and custom domains for Amazon Alexa, a popular commercial intelligent agent. To address scalability and data collection bottlenecks, we have proposed data efficient deep learning architectures that benefit from transfer learning from resource-rich functionality domains. Our models are pre-trained on existing resources and then adapted to hundreds of new, low resource tasks, allowing for rapid and accurate expansion of NLU functionality. In the future, we plan to explore unsupervised methods for transfer learning and the effect of semantic similarity between source and target tasks.

## References

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jan Buys and Jan A. Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *Proceedings of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6045–6049. IEEE.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning,*.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of International Conference of Machine Learning (ICML) 2008*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

DialogFlow. https://dialogflow.com.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271.

J.-K. Kim, Y.-B. Kim, R. Sarikaya, and E. Fosler-Lussier. 2017a. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2832–2838.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017b. Adversarial adaptation of synthetic or stale data. In *ACL*.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017c. Domain attention with an ensemble of experts. In *Annual Meeting of the Association for Computational Linguistics*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Björn Hoffmeister, Markus Dreyer, Stanislav Peshterliev, Ankur Gandhe, Denis Filiminov, Ariya Rastrow, Christian Monson, and Agnika Kumar. 2017. Just ASK: building an architecture for extensible self-service spoken language understanding. In *NIPS 2017 Workshop on Conversational AI*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Q. Li, G. Tur, D. Hakkani-Tur, X. Li, T. Paek, A. Gunawardana, and C. Quirk. 2014. Distributed open-domain conversational understanding framework with domain independent extractors. In *Spoken Language Technology Workshop (SLT) 2014*.

LUIS. The Microsoft Language Understanding Intelligent Service (LUIS), https://www.luis.ai.

L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin. 2016. How transferable are neural networks in nlp applications. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 479–489.

D. Nadeau and S. Sekine. 2007. *A survey of named entity recognition and classification. Linguisticae Investigationes,*, volume 1. John Benjamins Publishing Company.

W. Radford, X. Carreras, and J. Henderson. 2015. Named entity recognition with document-specific kb tag gazetteers. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 512–517.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 231–235.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.