Bootstrapping Chatbots for Novel Domains

Petr Babkin, Md Faisal Mahbub Chowdhury, Alfio Gliozzo, Martin Hirzel, Avraham Shinnar
IBM Research AI
Yorktown Heights, NY
{petr.babkin, mchowdh, gliozzo, hirzel, shinnar}@us.ibm.com

Abstract

We tackle the problem of automatically generating chatbots from Web API specifications using embedded natural language metadata, focusing on the intent classification subtask. One of the main challenges for such a use case comes from the lack of a sufficiently representative training sample for utterance classification, which hinders the traditional supervised model's ability to generalize to unseen inputs. We apply several unsupervised and distantly-supervised techniques to refine the model's representation and to augment its coverage. These include sentence similarity estimation in the embedding space and decision tree learner-based feature selection leveraging structural regularities in API specifications. In addition, we harness linguistic resources such as web corpora, PPDB, and WordNet. We use the resulting representation for prediction and perform a small-scale evaluation, where our approach compares favorably to the supervised baseline.

1 Introduction

Conversational bots serving Web APIs are becoming ubiquitous as nearly every major eCommerce company nowadays employs a chatbot front-end to their services. In turn, virtually every major player in the AI arena offers a product to support this [1, 2, 10, 11]. But with this interesting use case comes the technical challenge of scaling the dialogue and language models to novel APIs with minimal hand-labeling, feature engineering, and re-training for each individual domain.

Recently, there has been much progress in developing conversational systems based, most notably, on generative sequence-to-sequence recurrent architectures [7, 13, 21, 24]. Alas, these approaches require dialogue corpora (sets of question-answer or utterance-logical form pairs) to train them, which are not always available for an arbitrary domain of interest. That becomes a serious limitation when the dialogue domain is not pre-defined but is specified dynamically, e.g., by a supplied Web API [23]. While the dynamic generation of rudimentary chatbots is feasible, they come nowhere close in flexibility and robustness to models trained on large volumes of data. For the purpose of this paper we will focus on the intent classification subtask. Traditionally, the lack of training data for an NLP task is addressed by resorting to unsupervised and weakly supervised methods such as expectation-maximization and instance-based learning. Various techniques such as bootstrapping have been successfully employed on top, in some cases matching the performance of fully-supervised methods [6]. In recent years, pre-trained word embeddings have become a standard to improving the robustness of models trained on limited amounts of data on a wide array of NLP tasks, including sentence classification [12]. Another option that gained much popularity in the recent decade is transfer learning-based domain adaptation [17]. In this paper we pursue several of these techniques, contributing a novel feature selection method to the mix.

2 The Domain

The data at our disposal is a collection of 566 Swagger [16] files describing publicly available Web APIs for services ranging from language translation to ride sharing. Internally, each Swagger is a

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

JSON-formatted specification of available endpoints, methods, parameters and return values. While the specification is intended to be machine-readable, it does contain natural-language annotation fields such as descriptions. Figure 1 is an excerpt detailing the interface of the language identification endpoint from a language-translation API [9].

```
1  "swagger": "2.0",
2  "info": {"version": "2.0.0", "title": "Language Translator"}, ...
3  "basePath": "/language-translator/api",
4  "paths": {
5    "/v2/models": {"get": ..., "post": ...},
6    "/v2/models/{model_id}": {"get": ..., "delete": ...},
7    "/v2/translate": {"get": ...},
8    "/v2/identifiable_languages": {"get": ...},
9    "/v2/identify": {"get": {
10     "operationId": "identifyLanguageGet",
11     "summary": "Identifies the language of the input text",
12     "description": "The return value is the two-letter language code for the identified language.",
13     "parameters": [{
14       "name": "text",
15       "description": "Input text in UTF-8 format."}],
16     "responses": {"200": {
17       "identifiedLanguages": {
18         "description": "A ranking of identified languages with confidence scores.",
19         "type": "array",
20         "items": {
21           "identifiedLanguage": {
22             "language": {
23               "description": "The code for an identified language.",
24               "type": "string" },
25             "confidence": {
26               "description": "The confidence score for the identified language.",
27               "type": "number" }, ...
```

Figure 1: Excerpt from the Translation API Swagger.

3 Approach

Our baseline is a proprietary compiler [23] that generates dialogue workspaces for Watson Conversation Service (WCS) [10]. It deterministically extracts endpoint and parameter names from Swagger and supplies them as training examples to the WCS classifiers for *intents* and *entities*, respectively.

While the resulting dialogue system is able to classify user utterances with reasonable accuracy, it only understands a limited vocabulary and forces the user to use stylized language where the input words have to match the keywords it learned. This limitation makes for unnatural user interaction. For example, consider the verbose utterance (a) supported by the system and its more implicit equivalent (b) that is beyond the capability of the underlying ngram-based model.

identify the language of text 'vive' (a)
is vive **spanish**? (b)

The underlined words in (a) match the keywords extracted from Lines 9 and 14 of the Swagger in Figure 1, reflecting the names of the service endpoint (identify) and its required parameter (text). In contrast, the utterance (b) contains none of the overt keywords and requires subtle inference that the word spanish is related to language. Furthermore, the baseline model is insensitive to syntax, allowing utterances to match a wrong command based on word overlap, as in utterance (c):

what languages **can** you identify? (c)

While Swagger files contain more information, using it effectively is challenging. First, the amount of textual data describing each endpoint is prohibitively low to be used for training directly, as the resulting model will simply overfit to linguistic idiosyncrasies of individual examples rather than learning useful generalizations pertaining to underlying intents. Second, different endpoints within the same API are topically related and thus share much of the vocabulary (e.g., in the Translation API, the description of most endpoints involves the word "language", and in the Lyft ride sharing

API, many commands have to do with rides). The shared vocabulary makes topic modeling-based approaches such as LSI [4] infeasible, which warrants the use of a more fine-grained representation.

To combat the data deficit we use word embeddings and a parsing model pre-trained on large corpora [8]. In addition, we bootstrap our model with information from external linguistic resources such as PPDB [5] and WordNet [14]. Finally, we opt for a finer-grained feature representation that exploits structural regularities in Swagger. The following sections detail the specifics of our approach.

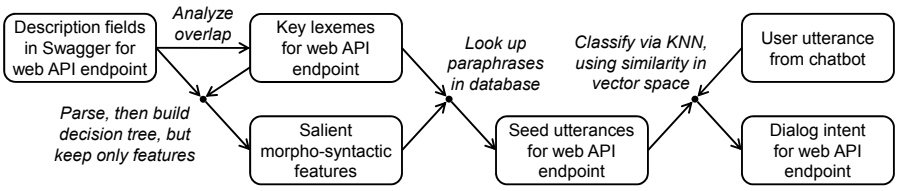

Figure 2: Overview of our approach

3.1 Estimation of Semantic Similarity under Vector Space Model

The first step to making our model more robust is to abstract away from discrete string matching. This can be accomplished by introducing word embeddings, which enable a fuzzier notion of *semantic relatedness* under vector space model (VSM). Thus, we can replace the binary notion of whether an item matches a certain keyword with a continuous similarity measure given by the cosine of the angle between the corresponding dense vectors. We used 300-dimensional vectors pre-trained with the GloVe algorithm [19] on the Common Crawl corpus that come with the gensim Python library [20]. By applying this model, for the words "spanish" and "language" we get a significant similarity score close to 0.4 (for reference, the similarity between "spanish" and "english" is around 0.8 and between "language" and "languages" is close to 0.9 under the same model, whereas the similarity among unrelated words is usually below 0.1)

Next comes a question of how to compute similarity among clauses and complete sentences. There are a number of approaches to embedding composition including averaging of individual word embeddings, their weighted addition, multiplication, dilation, circular convolution (cf. [15] for a good overview) and even subtraction [3]. Some more advanced, syntax-aware approaches include compositional vector grammars [22] and recursive tree autoencoders [25]. After experimenting with a few different models we found the following simple similarity metric works best:

$$sim(\vec{u}, \vec{s}) = \sum_i \max_j \cos(\vec{u}_i, \vec{s}_j) \tag{1}$$

where vectors $\vec{u}$ (user utterance) and $\vec{s}$ (seed utterance) represent sentences such that their ith or jth component is a GloVe embedding of the ith or jth word, respectively. This equation is not symmetric in that it sums over the component embeddings of the user utterance, for each aiming to find the closest component of the seed utterance. This is motivated by the need to assign higher scores to those seeds that match not only based on intent (as in simple max or averaging) but also on arguments. We use this metric in K-nearest neighbors classification of incoming user utterances.

3.2 Selecting Optimal Representation

Computing the similarity from raw annotations without preprocessing resulted in suboptimal performance. When using a VSM, we want to make sure that only the invariant features fundamentally characteristic of the underlying intent are considered while mere wording idyosyncracies are ignored. The approach we took with respect to selecting an optimal representation was a) to determine salient lexemes and their salient morpho-syntactic configurations, and b) reduce raw annotations to "compressed" form while preserving their discriminative attributes. A simple way to identify potentially salient words is to look for lexemes that overlap across different fields describing the same endpoint.

The lemmas "language" and "identify" are most prevalent in the /v2/identify endpoint. However, one should be wary that due to substantial lexical overlap across classes and the low volume of available data, lexical features alone might not suffice. Thus, we can consider the key lexemes in conjunction with their functional role in the sentence, which are likely to be different across classes.

Table 1: Lexically overlapping but functionally distinct descriptors of two different endpoints.

/v2/identify	/v2/identifiable_languages
VBZ dobj NN **identifies** the **language** of the input text	NNS relcl VBN lists all **languages** that can be **identified** by the api
VBN amod NNS a ranking of **identified languages** with confidence scores	NNS relcl VB a list of all **languages** that the service can **identify**
VBN amod NN a code for an **identified language**	JJ amod NN the code for an **identifiable language**
VBN amod NN the confidence score for the **identified language**	JJ amod NN the name of the **identifiable language**

For example, consider syntactic dependencies indicated in Table 1. The same word that acts as a head verb (identifies) or a verbal modifier (identified) in the descriptions of the first endpoint, takes on the adjectival modifier (identifiable) and the relative clause verb roles in texts describing the second one. We capture these fine-grained distinctions by decomposing each key lexeme into its base form, morphological features (e.g., part of speech, number, etc.), and syntactic features (e.g., dependency relation, subordinate and superordinate constituents). Next, to select the most salient subset of features we employ the information gain-based decision tree algorithm provided by the *sklearn* library [18]. To illustrate, Table 2 shows the two features selected by this procedure in order to perfectly discriminate between the two classes in the example. Importantly, the resulting feature hierarchy is not used for classification but rather as a "filter": stripping down the seed examples to key lexemes exhibiting feature configurations that it identified to be salient to each class.

Table 2: The resulting salient feature space w.r.t. the lexeme "identify" obtained for the two classes.

	verb (VB, VBZ, VBN)	adjective (JJ)
adjectival modifier (amod)	/v2/identify	/v2/identifiable_languages
direct object (dobj)	/v2/identify	/v2/identifiable_languages
relative clause (relcl)	/v2/identifiable_languages	/v2/identifiable_languages

3.3 Data Augmentation and Boosting

After the seed utterances have been reduced to their minimal representations in terms of the identified set of discriminative features, we enrich them using external linguistic resources, namely the paraphrase database (PPDB) [5] to widen the system's coverage. PPDB was collected with a high precision method relying on parallel corpora and statistical co-occurence measures, which ensures high quality paraphrases. Functionally, this database is not simply a thesaurus of synonyms but also includes phrasal and syntactic paraphrases. For example, using PPDB, a clause "give me a ride" from the ride sharing domain can be paraphrased in multiple interesting ways including a subtle "pick me up". In addition, we conducted a limited series of trials with incorporating user feedback — by adding correctly labeled utterances to the pool of seed examples as a form of boosting our classifier. However, we observed the impact on the performance to be too unstable and, due to the limited time for the project, did not pursue this direction further.

4 Results

We performed a small-scale evaluation on three different Web API Swaggers — Watson Translation, Lyft, and Petstore. For each, we elicited approximately 30 – 40 utterances from 4 annotators who use Web API Swaggers in their work. They were encouraged to provide concise yet informal commands to the bot as implicit as they would to a friend in a chat. As part of the annotation, they indicated the intended endpoint (from the list in the API Swaggers) for every typed utterance. Agreement scores were not formally measured.

As expected, simple synonymy and lexical paraphrase got picked up easily by the system. For example, "remove model" in the translation API was correctly recognized as endpoint "delete

translation model", and "abort mission to new york" from the Lyft API was resolved to the "cancel ride" endpoint. A significant fraction of more implicit renderings that do not contain command identifying verbs were also correctly recognized; for example, "what language is this: saionara?" as "identify language" and "i don't want this ride" as "cancel ride". Interestingly, in such subtle cases, certain stop words such as propositions came out as crucial: "guten tag from german to english" (translate) as the seed descriptions contained source and destination languages to translate "from" and "to". In addition, type vs. token distinction played a role as language names were found similar to the word "language".

However, there is a limit to the semantic subtlety that our approach can handle. For example, a very open ended "what languages do you speak anyway?" utterance (intended: list identifiable languages) and even more implicit ones such as "i wanna go somewhere else" (intended: "change destination") require more sophisticated semantic reasoning going beyond sheer similarity. Quantitatively, the accuracies varied greatly across domains. Nevertheless, our approach consistently improved over the baseline (which is described in Section 3). Figure 3 shows an absolute improvement over the baseline of 0.33 for Translation and 0.14 for Lyft. Judging the improvement on the Petstore test set proved difficult due to a lack of reliable numbers of baseline performance (the baseline compiler wrongly merged multiple intents together in this Swagger). The absolute accuracy of 0.44 (across 19 classes), while not ideal, is still significantly better than random ($1/19 \approx 0.05$).

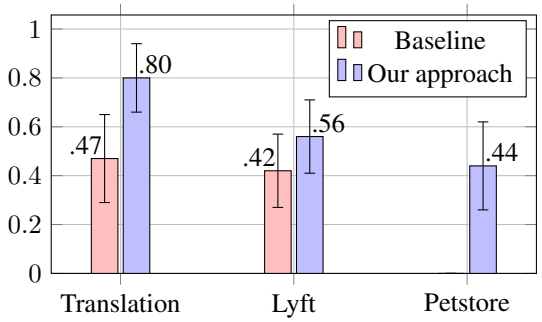

Figure 3: Classification accuracy comparison

In addition, for the Translation domain, we compared against the performance on raw (i.e. unprocessed) utterances, which as one would expect resulted in performance not only inferior to our main approach but even slightly worse than the deterministic baseline (0.37). Although we did not perform a formal ablation study, the runner-up configuration (max similarity and no PPDB) achieved 0.67 score. Examination of confusion matrices revealed that most errors occur between semantically similar classes, e.g., "get rides" and "get ride types". Among miscellaneous observations, "give rating" was sometimes confused with "cancel ride" probably in part based on the negative sentiment of the corresponding utterance such as "very bad driver, 0 stars" that was mistakenly considered as related to a slightly negatively charged word "cancel". In addition, utterances invoking "get cost estimate" such as "how much for the ride", or "how much do i pay" were confused with a broad variety of endpoints whose corresponding utterances also contain question words, for example: "when will my ride arrive" (get ETA), "who are the passengers" (get profile), and "what types of rides are available from this location" (get ride types).

5 Conclusion

Learning NLP models from very limited training data remains a challenge that does not have a universal solution. In this project we experimented with multiple unsupervised and weakly supervised techniques and, in addition, harnessed our knowledge of the nature and format of our data. We achieved moderate success in a hard problem of multiclass sentence classification with just a handful of training examples. In particular, we found representation augmentation in combination with automatic feature selection to be especially beneficial in a limited data setting. But while our approach worked really well in one domain, this did not trivially transfer to others. This underscores the importance of a reliable proxy metric for estimating and tuning model performance prior to the evaluation on a labeled test set. We leave a formal investigation on that subtopic for future work in addition to other potentially fruitful avenues of training on noisy data and domain adaptation.

References

[1] Amazon. Lex, 2017. https://aws.amazon.com/lex/ (Retrieved November 2017).

[2] api.ai, 2010. https://api.ai/ (Retrieved November 2017).

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 2787–2795, 2013.

[4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764, 2013.

[6] Alfio Gliozzo, Carlo Strapparava, and Ido Dagan. Improving text categorization bootstrapping via unsupervised learning. *Transactions on Speech and Language Processing (TSLP)*, 6(1):1:1–1:24, October 2009.

[7] 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 *Conference of the International Speech Communication Association (INTERSPEECH)*, pages 715–719, 2016.

[8] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1373–1378, 2015.

[9] IBM. Watson Language Translator service, 2015. https://www.ibm.com/watson/developercloud/language-translator.html (Retrieved October 2017).

[10] IBM. Watson Conversation service, 2016. https://www.ibm.com/watson/developercloud/conversation.html (Retrieved October 2017).

[11] kitt.ai, 2016. https://kitt.ai (Retrieved November 2017).

[12] Amit Mandelbaum and Adi Shalev. Word embeddings and their use in sentence classification tasks, 2016. https://arxiv.org/abs/1610.08229.

[13] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. Using recurrent neural networks for slot filling in spoken language understanding. *Transactions on Audio, Speech, and Language Processing (TASLP)*, 23(3):530–539, March 2015.

[14] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM (CACM)*, 38(11):39–41, November 1995.

[15] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 236–244, 2008.

[16] OpenAPI Initiative. OpenAPI specification (fka swagger restful api documentation specification), 2014. https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md (Retrieved November 2017).

[17] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Transactions on Knowledge and Data Engineering (ToKaDE)*, 22:1345–1359, October 2010.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[20] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.

[21] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808, 2015.

[22] Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. Parsing with compositional vector grammars. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 455–465, 2013.

[23] Mandana Vaziri, Louis Mandel, Avraham Shinnar, Jérôme Siméon, and Martin Hirzel. Generating chat bots from web API specifications. In *Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, pages 44–57, 2017.

[24] Oriol Vinyals and Quoc V. Le. A neural conversational model. In *Deep Learning Workshop*, 2015.

[25] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In *International Conference on Machine Learning (ICML)*, pages 1604–1612, 2015.