EDITOR: Erik Cambria, Nanyang Technological University

Intent Classification for Dialogue Utterances

Jetze Schuurmans
Flavius Frasincar
Erasmus University Rotterdam

In this work we investigate several machine learning methods to tackle the problem of intent classification for dialogue utterances. We start with *Bag-of-Words* (BoW) in combination with *Naïve Bayes* (NB). After that, we employ *Continuous Bag-of-Words* (CBoW) coupled with *Support Vector Machines* (SVM). Then follow *Long Short-Term Memory* (LSTM) networks, which are made bidirectional. The best performing model is hierarchical, such that it can take advantage of the natural taxonomy within classes. The main experiments are a comparison between these methods on an open sourced academic dataset. In the first experiment we consider the full dataset. We also consider the given subsets of data separately, in order to compare our results with state-of-the-art vendor solutions. In general we find that the SVM models outperform the LSTM models. The former models achieve the highest macro-F1 for the full dataset, and in most of the individual datasets. We also found out that the incorporation of the hierarchical structure in the intents improves the performance.

Customer interaction is at the center of many organizations. In order to help customers efficiently, one could automate the interaction between the organization's representative and a customer. Customers usually contact the organization with a specific request or query. In order to help a customer, the intention of the customer needs to be classified.[1] Intent classification tries to answer the question why the customer contacted the organization and what the customer wants to achieve. The interaction can partly or fully be automated using a dialogue system[2], which uses intent classification. The classification can also be used to help the human representatives, namely, by using intent classification to direct the incoming messages to the representative that has the right expertise. Due to its importance for dialogue handling[3], intent classification needs to be done properly. Therefore, this research focuses on improving the existing practice of intent classification for dialogue utterances.

In order to classify intents of customers, a dialogue system needs to analyze the incoming messages. The messages are called utterances, or acts-of-speech. In our case they are typed messages in English, roughly the length of a sentence. The classification of the intent is made per utterance. We analyze the case where possible intents are disjoint. In other words, each incoming message belongs to only one class. However, some intents might be very similar and belong to a common category, or in other words to a group of intents. We explore the possibility of extending the classifier with knowledge about the inherent hierarchy of intents.

RELATED WORK AND SCIENTIFIC RELEVANCE

Previous studies have proposed several classification algorithms for short texts, starting with parsimonious text classifiers, such as BoW with NB and CBoW with SVM.[4] The performance of NB is limited by the vocabulary in the training set. SVM can circumvent this by using word embeddings, trained on an external corpus. However, with both approaches, word order is lost. To account for complex dependencies between words in the representation of an utterance *Recurrent Neural Networks* were introduced.[5] Most recently, LSTMs and their simplification Gated Recurrent Unit (GRU) have been used for intent classification[6] and emotion detection[7], respectively, in dialogues. Attentive LSTMs[8] are less useful here as the classified text is rather short in nature.

Flat classifiers need to distinguish between all classes at once. When there is a large number of classes, this can become difficult. Instead, *hierarchical classification* can be used. A hierarchical classifier tries to incorporate the hierarchical structure of the class taxonomy. Hierarchical classification was first used for text classification by Koller and Sahami.[9] They used a local classifier per parent node for training, at each node selecting a subset of features relevant for that step in the classification process. A similar hierarchical structure with an SVM at every node was used for speech-act classification.[10] Ono et al. used a form of local classifier per level, where they tried the lowest level (leaf nodes) first.[11] If the uncertainty is too high, they move up in the hierarchical level. Hierarchical classifiers have been used for intent classification in Web[12] and platform[13] searches. For chatbots multi-intent classification was researched by Rychalska et al.[14]

We contribute to the existing literature in two ways. First, we apply hierarchical intent classification on dialogue utterances (in multi-class classification as apposed to multi-label). Secondly, we present performances of machine learning classifiers, alongside the black box models used by Braun et al.[15]

METHODOLOGY

In this section we discuss the methods used to classify intents. Each method is a combination of an utterance representation and a classification algorithm. We start with a formalization of the problem. Then follow the flat classifiers. Finally, we discuss the hierarchical classifier.

Intent Classification

The classification of an intent is answering the question: What is the customer trying to accomplish? In intent classification, the utterance $d \in X$ of a dialogue is given, where X is the utterance space; a fixed set of predefined intents $C = \{c_1, \ldots, c_J\}$; and a training set D of labeled dialogue utterances $\{d_i, c_i\}_{i=1}^{N}$, where $(d, c) \in X \times C$. We consider the one-off problem or in other words single label classification, where each d corresponds to one element of C. For example,

$(d, c) =$ (‘What software can I use to view epub documents?’, ‘Software Recommendation’).

Flat Classifiers

BOW-NB. The first model we discuss is the BoW representation with multinomial NB. This model is the baseline in our experiments. Each utterance is represented by the set of word counts that occur in the utterance. Therefore, word order is neglected. The way we implement NB is as follows. First, we start by removing the stop words. Secondly, we use lemmatization. Although the combination of uni-gram and bi-gram is advised[4], we do not have enough bi-gram counts. Therefore, we only use uni-grams. We handle zero counts with Laplace smoothing.

An advantage of NB is its efficiency during training time, as it only needs to pass through the data once. However, the downside of NB is the conditional independence assumption, stating that terms and the signal they carry are independent of each other given the class. Furthermore, the model uses the positional independence assumption, stating the position of a word does not matter. Most importantly NB cannot handle unseen words.

CBOW-SVM. Secondly, we discuss CBoW as an input for SVM. CBoW uses continuous word representations called word embeddings. This gives the SVM classifier the advantage to pick up signals from similar in meaning, yet unseen, words. We use three word embeddings: Word2Vec[16], GloVe[17], and FastText[18].

The CBoW representation is comparable to the conventional bag-of-words representation, since both lose the information of the order of terms. However, using word embeddings gives CBoW an advantage over traditional bag-of words. Namely, CBoW can pick up signals from previously unseen words. CBoW gives us the additional advantage that the input for the classification algorithm is a fixed dimensional vector, independent of the length of the utterance or vocabulary. This is a desirable feature for SVMs. There are two forms of CBoW we consider. One takes the sum of the embedding vectors of the respective terms, while the other takes the average:

$$CBoW_{sum}(t_1, \ldots, t_k) = \sum_{i=1}^{k} v(t_i), \quad CBoW_{ave}(t_1, \ldots, t_k) = \frac{1}{k} \sum_{i=1}^{k} v(t_i), \tag{1}$$

where each feature t_i corresponds to a word and has an associated vector $v(t_i)$.

The intuition behind CBoW is as follows. The summation of word vectors creates a path in the word embedding space. The resulting vector (from the origin to the end of the path) should capture a mathematical representation of the overall meaning of the utterance. Adding more words with the same meaning might spread the cluster of the representations of a given intent, possibly making the classification harder. When the average is taken, the overall length of this path is normalized with respect to the number of words in the utterance.

SVMs are a classification method that uses a kernel function to find decision boundary between two classes that has a maximum margin in a latent space. We consider both the *Linear* and *Radial Basis Function* kernels. Since we allow for misclassifications in the training set, a cost parameter C is added to give a penalty to these violations. In order to determine which kernel and hyperparameters to use, we use 2-fold cross validation with stratified sampling.

Inherently, SVMs are binary classifiers. Several attempts have been made to create a multiclass SVM scheme.[19] We use the one-against-one[20] scheme, as it performed as one of the best in the comparison of Hsu and Lin.[19] During testing we use Max Wins voting[21], where the class with the highest number of votes is chosen as final prediction. Since we are dealing with unbalanced class distributions, we use class weights in the SVM.

LSTM. The key feature of recurrent neural networks (RNN) is that they can process sequential data, giving them the possibility to model word dependencies. Parameter sharing enables the

recurrent network to pick up signals from longer sequences than dense neural networks, and to take inputs of arbitrary length and learn general patterns across them. There are several types of RNN architectures[22], we consider the *tail* model. The tail model constructs a hidden state by passing the complete sequence and using the last hidden state as input for the classification layer. Alternatives such as the *pooling* or *hybrid pooling* do not consistently outperform the more parsimonious tail model.[22]

Gated RNNs are the most compelling sequence models used in practice. These include networks based on the LSTM[23] and GRU[24]. Gated RNNs are based on the idea of creating paths through time that have derivatives that neither vanish nor explode. This is done by learning connection weights, and the ability to forget the old state, from the data. We choose to use LSTMs over GRUs due to the extra flexibility offered by the controls for the update and output of the state.

Bidirectional LSTM (BiLSTM) was created to model dependencies on the next time step in the sequence.[25] They are a combination of a recurrent module that passes the sequence forward through a memory block and a recurrent module that passes the sequence backwards through a different memory block. The tail model uses a concatenation of the final two hidden states as input for the last layer.

Following similar work the network is trained using the Adam optimizer.[6] We calculate updates from the gradients based on batches of training utterances. We use Back-Propagation Through Time (BPTT)[26] to update recurrent components. Gradient Clipping is used in order to deal with exploding gradients and we found that capping the gradients at 5 works well. We use the following regularizers: early stopping, ensembles, and weight noise. A popular way of creating weight noise is by applying dropout. We use dropout only at the non-recurrent connections.[27] The hyperparameters of the LSTM model are the size of the input dimension, and the size of the state variable. Both are determined by 2-fold cross validation using stratified sampling.

Hierarchical Classifiers

Hierarchical classification can be considered as a classification that takes the hierarchical structure of the taxonomy of classes into account, as opposed to a flat classifier, which only takes the final classes into account. By imposing the hierarchical structure, the model does not need to learn the separation between a large number of classes. It can now focus on classifying subclasses within a category. The taxonomy can be formalized as a *tree* or a *Directed Acyclical Graph*[28], we consider the case where the taxonomy is a tree due to the nature of our data.

Our goal is to reduce the number of classes considered based on the natural taxonomy, therefore we use a local classifier per parent node. This local hierarchical classifier has a flat classifier at every parent node, which means that the number of classifiers that need to be constructed scales directly with the number of parent nodes. During training of a classifier at any given parent node, only the observations belonging to its children are considered. After training each individual classifier, the local classifier can be used for inference. During testing, the classification starts at the root node. The outcome of the root node determines which next classifier should be considered. The outcome of this classifier selects the next classifier to be used. This is repeated until a leaf node is predicted, this then becomes the final prediction of the local classifier.

Performance Measure

We measure the performance with the macro-F1 score. The F1 score is a harmonic mean of the precision and recall for each intent. We value both and do not want a linear trade-off between them. We are interested in the performance on all classes equally, independent of the number of test observations. Therefore, we aggregate the measures by means of the macro average.

DATASET AND EXPERIMENTS

We use the dataset curated by Braun et al.[15], available at https://github.com/sebischair. It consists of two corpora, distinguished by the way they were gathered. There is the Chatbot Corpus on Travel Scheduling, and the StackExchange Corpus on Ask Ubuntu and Web Applications. In this section we discuss the experimental setups on this dataset.

Complete Dataset. We start with the complete set that includes all training observations and all intents. This gives us the opportunity to select the best overall model. The best model is selected based on the macro-F1 score. The concatenation of the three subsets imposes a hierarchy in the taxonomy of intents. This allows us to compare hierarchical classifiers with flat classifiers. The class hierarchy and the local classifiers are depicted in Figure 1.

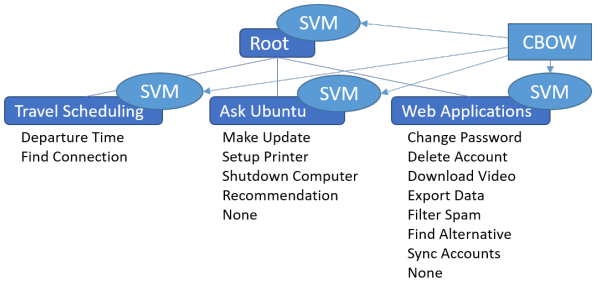

Figure 1. Hierarchy of the classes with a local hierarchical classifier per parent node.

Individual Datasets. In this experiment we consider the subsets of the data separately. This gives us the possibility to compare our methods with the classifiers used by Braun et al.[15] They use the Natural Language Understanding solutions of LUIS, Watson Conversation, API.ai, and RASA.

RESULTS

Complete Dataset. The results of the different classifiers on the complete dataset are reported in Table 1. The best performing flat classifier is the SVM model, this is independent of the type of word embedding or the method used to aggregate the word embeddings. We select this classifier as candidate for the hierarchical classifier. When adding hierarchy to the models, we find varying results. The baseline model clearly improves when taking the taxonomy of classes into account, while adding the local hierarchy to the SVMs comes with mixed results. For the FastText embeddings it is a clear improvement, while for the GloVe embeddings it is not. Overall, the best hierarchical SVM outperforms the best flat SVM.

Table 1. Macro-F1 for the test set on the complete dataset.

Model	Flat	Hierarchical	Model	Flat
NB	.541	.614		
SVM FastText average	.689	**.782**	LSTM FastText	.605
SVM FastText sum	.657	.642	BiLSTM FastText	.569
SVM GloVe average	**.752**	.654	LSTM GloVe	.586
SVM GloVe sum	.680	.658	BiLSTM GloVe	.575
SVM Word2Vec average	.705	.703	LSTM Word2Vec	.543
SVM Word2Vec sum	.673	.706	BiLSTM Word2Vec	.502

With regard to the utterance representation we find that averaging is better than summing the word embeddings, as SVM with $CBoW_{ave}$ performs better in the flat classification and the best hierarchical classifier uses also averages. Furthermore we note that the bidirectional component in the BiLSTMs does not capture more information, as the LSTM performs better than the BiLSTM. Together with the fact that the SVM outperforms the LSTM, this indicates that taking the word order into account is not relevant in this dataset. This is likely due to the short utterance length.

Individual Datasets. The macro-F1 scores for the individual datasets are presented in Table 2. We note that it is hard to interpret the comparison with Braun et al.[15], as most of the methods used are black boxes.

Table 2. Macro-F1 score for the individual subsets.

	Travel Scheduling	Ask Ubuntu	Web Applications
NB	.959	.726	.502
SVM FastText average	.958	**.812**	**.771**
SVM FastText sum	.968	.800	.658
SVM GloVe average	.946	.805	.591
SVM GloVe sum	.957	.729	.692
SVM Word2Vec average	.979	.742	.698
SVM Word2Vec sum	.946	.742	.680
LSTM FastText	.968	.644	.465
BiLSTM FastText	.979	.646	.549
LSTM GloVe	.945	.665	.546
BiLSTM GloVe	.979	.667	.635
LSTM Word2Vec	**.989**	.631	.395
BiLSTM Word2Vec	**.989**	.710	.443
LUIS	**.979**	.743	**.690**
Watson	.968	**.819**	.630
API.ai	.931	.782	.628
RASA	**.979**	.708	.494

In the Travel Scheduling dataset, the (Bi)LSTM with Word2Vec embeddings performs the best. The SVM with Word2Vec and $CBoW_{ave}$ and BiLSTM perform equally well as the intent classifiers of LUIS and RASA. We note that the relatively high performance of our baseline, NB, indicates that this is a relatively easy set to classify.

The Ask Ubuntu set provides a slightly harder classification task. In this set the intent classifier of Watson outperforms the other vendor solutions as well as all our models. From our models, the SVM with FastText with $CBoW_{ave}$ is the best performing model. We note that all recurrent neural networks are performing worse than the NB baseline.

The final subset is on Web Applications. The Web Applications data proves to be more difficult, this is likely due to the fact that it has very few training observations (an average of less than 4 training observations per intent). Here we see that our best performing model is the SVM with FastText and $CBoW_{ave}$. Together with the Word2Vec $CBoW_{ave}$ and the GloVe $CBoW_{sum}$ it outperforms the vendor solutions. Furthermore, we note that the BiLSTM is the best performing recurrent network, just as in the Travel Scheduling and Ask Ubuntu sets. One can note that on the complete dataset the LSTM performed better than BiLSTM, as the LSTM has an edge in differentiating between the three types of datasets.

CONCLUSION

In general we find that the SVM models outperform the LSTM models. They achieve the highest macro-F1 for the full dataset, as well as the ability to handle the scenario of the individual datasets. With regard to taking advantage of the hierarchical structure in the intents, we find that the SVM with averaged FastText embeddings significantly benefits from the hierarchy and outperforms all other models. Using word embeddings as utterance representation yields a better performance than using a count based method. However, taking word order into account does not. In general we see better results when we take the element wise average of the word embeddings, as apposed to the sum, indicating that correcting for the length of the utterance is useful. Finally, we note that our models improve on the NB baseline. Furthermore, they are on par with or improve on the performance of the black box methods used by Braun et al. [15]

Future Research

There are different opportunities for future work, we discuss a few below. We start with several options with respect to the hierarchy, followed by data augmentation and transfer learning.

The type of hierarchical model considered is a local hierarchical classifier per parent node. Alternatively a global hierarchical classifier could be constructed by modifying a flat classifier to take the taxonomy into account at once. The intermediate certainties could be exploited by the dialogue system, with specific follow-up questions.

In order to deal with the limited number of training observations, future work could look into data augmentation or transfer learning. Data augmentation could be used by interchanging one or multiple random words with their synonyms. Alternatively, transfer learning can be used. One could take a subset of intents, starting with 2 intents, training the classifier and using the inferred weights as initialization when learning to classify with an additional intent added to the problem.

REFERENCES

1. N. Howard and E. Cambria, “Intention awareness: Improving upon situation awareness in human-centric environments,” *Human-centric Computing and Information Sciences*, vol. 3, no. 1, 2013, p. 9.
2. T. Young et al., “Augmenting end-to-end dialogue systems with commonsense knowledge,” *Proc. of the Thirty-Second AAAI Conf. on Artificial Intelligence (AAAI-18)*, AAAI Press, 2018, pp. 4970–4977.
3. C. Welch et al., “Learning from personal longitudinal dialog data,” *IEEE Intelligent Systems*, vol. 34, no. 4, 2019, pp. 16–23.
4. S. Wang and C. D. Manning, “Baselines and bigrams: Simple, good sentiment and topic classification,” *Proc. of the 50th Ann. Meeting of the Assoc. for Computational Linguistics (ACL-12)*, 2012, pp. 90–94.
5. M. Henderson, B. Thomson, and S. Young, “Word-based dialog state tracking with recurrent neural networks,” *Proc. of the 15th Ann. Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL-14)*, 2014, pp. 292–299.
6. L. Meng and M. Huang, “Dialogue intent classification with long short-term memory networks,” *Proc. of the National CCF Conf. on Natural Language Processing and Chinese Computing (NLPCC-19)*, 2017, pp. 42–50.
7. N. Majumder et al., “DialogueRNN: An attentive RNN for emotion detection in conversations,” *Proc. of the Thirty-Third AAAI Conf. on Artificial Intelligence (AAAI-19), vol. 33*, AAAI Press, 2019, pp. 6818–6825.

8. D. Hu, "An introductory survey on attention mechanisms in NLP problems," *2019 Intelligent Systems Conference (IntelliSys-19),* Advances in Intelligent Systems and Computing*, vol. 1038*, Springer, 2019, pp. 432–448.
9. D. Koller and M. Sahami, *Hierarchically classifying documents using very few words*, Tech. rep., Stanford InfoLab, 1997.
10. S. Kang, Y. Ko, and J. Seo, "Hierarchical speech-act classification for discourse analysis," *Pattern Recognition Letters*, vol. 34, no. 10, 2013, pp. 1119–1124.
11. K. Ono et al., "Toward lexical acquisition during dialogues through implicit confirmation for closed-domain chatbots," *Proc. of the Workshop on Chatbots and Conversational Agent Technologies (WOCHAT-16)*, 2016.
12. B. J. Jansen, D. L. Booth, and A. Spink, "Determining the informational, navigational, and transactional intent of Web queries," *Information Processing & Management*, vol. 44, no. 3, 2008, pp. 1251–1266.
13. J. Hu et al., "Understanding user's query intent with Wikipedia," *Proc. of the 18th Int'l Conf. on World Wide Web (WWW-09)*, ACM, 2009, pp. 471–480.
14. B. Rychalska, H. Glabska, and A. Wroblewska, "Multi-intent hierarchical natural language understanding for chatbots," *Proc. of the Fifth Int'l Conf. on Social Networks Analysis, Management and Security (SNAMS-18)*, 2018, pp. 256–259.
15. D. Braun et al., "Evaluating natural language understanding services for conversational question answering systems," *Proc. of the 18th Ann. SIGdial Meeting on Discourse and Dialogue (SIGDIAL-17)*, 2017, pp. 174–185.
16. T. Mikolov et al., "Distributed representations of words and phrases and their compositionality," *Proc. of the 27th Ann. Conf. on Neural Information Processing Systems (NIPS-13)*, 2013, pp. 3111–3119.
17. J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP-14)*, ACL, 2014, pp. 1532–1543.
18. P. Bojanowski et al., "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, 2017, pp. 135–146.
19. C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, 2002, pp. 415–425.
20. S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," Neurocomputing, Springer, 1990, pp. 41–50.
21. J. H. Friedman, "Another approach to polychotomous classification," *Technical Report, Statistics Department, Stanford University*, 1996.
22. L. Shen and J. Zhang, "Empirical evaluation of RNN architectures on sentence classification task," *arXiv, https://arxiv.org/abs/1609.09171*, 2016.
23. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, 1997, pp. 1735–1780.
24. K. Cho et al., "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.
25. A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, 2005, p. 602.
26. P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, 1990, pp. 1550–1560.
27. W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv, https://arxiv.org/abs/1409.2329*, 2014.
28. C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, 2011, pp. 31–72.

ABOUT THE AUTHORS

Jetze Schuurmans is a graduated Master of Science student at Erasmus University Rotterdam, Erasmus School of Economics. He is currently working for CaptainAI, enabling autonomous shipping with AI. Contact him at jetzeschuurmans@gmail.com.

Flavius Frasincar is an assistant professor at Erasmus University Rotterdam. His research interests lie in Web information systems, personalization, machine learning, and the Semantic Web. He was awarded a PhD in information systems from Eindhoven University of Technology. Contact him at frasincar@ese.eur.nl.