Detecting Interrogative Utterances
with Recurrent Neural Networks

Junyoung Chung[*]
Université de Montréal
Montréal, Canada
junyoung.chung@umontreal.ca

Jacob Devlin   Hany Hassan Awadalla
Microsoft Research
Redmond, USA
{jdevlin, hanyh}@microsoft.com

Abstract

In this paper, we explore different neural network architectures that can predict if a speaker of a given utterance is asking a question or making a statement. We compare the outcomes of regularization methods that are popularly used to train deep neural networks and study how different *context* functions can affect the classification performance. We also compare the efficacy of gated activation functions that are favorably used in recurrent neural networks and study how to combine multimodal inputs. We evaluate our models on two multimodal datasets: MSR-Skype and CALLHOME.

1   Introduction

Spoken language understanding is a long-term goal of machine learning and potentially has a huge impact in practical applications. However, the difficulty of processing speech signals itself is a bottleneck, for instance, the core part of speech translation has to be processed in the text domain. In other words, a failure of capturing the key features in the speech signals can lead the next applications into unexpected results.

Identifying whether a given utterance is a question or not can be one of the key features in applications such as speech translation. Unfortunately, a speech recognition system is likely to fail achieving two goals at a same time: (1) extract text sequences from the input utterances, (2) detect questions. We can think of a question detection system that works independently and unburdens the load of the speech recognition system [15, 10, 4, 14, 3]. Later, an annotation of being a question can form a set with the output of the speech recognition system and handed over to the machine translation system.

Previous studies have focused on using hand-designed features and classifiers such as support vector machines (SVMs) [10, 14] or tree-based classifiers [15, 4, 3]. The classifiers used in these systems are shallow and simple, but there are considerable efforts on designing features based on domain knowledge. However, there is no guarantee that these hand-designed features are optimal to solve the problem.

In this study, we will let the model to learn the features from the training examples and the objective function. We propose a recurrent neural network (RNN) based system with various model architectures that can detect questions using multimodal inputs. Our question detection system runs as fast as other real-time systems at the test time, receives multimodal inputs and returns a scalar score value $\hat{y} \in [0, 1]$. We evaluate our models on two multimodal datasets, which consist with pairs of text transcripts and audio signals. Our experiments reveal what types of *context* functions, regularization methods, state transition functions of RNNs and data domains are helpful in RNN-based question detection systems.

---

[*]Work done while the author was at Microsoft.

1

Table 1: Types of questions

	Examples
Yes-No	Did you attend the meeting?
wh-words	Where have you been?
Declarative	You are at the meeting?

2 Background

2.1 Types of Questions

Questions can have different canonical forms, and they are usually not standardized. However, we can divide the questions into three groups based upon some criteria. Table 1 shows an example from each group. We note that declarative questions are rather unclear to differentiate from non-question statements by looking into their canonical forms because they usually do not contain any *wh*-words. However, audio signals might contain the features that can be useful when making predictions on this kind of examples, where a question usually contains a rising pitch at the end of the utterance.

2.2 Neural Networks

An RNN can process a sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ by recursively applying a transition function g to each symbol:

$$\mathbf{s}_t = g(\mathbf{x}_t, \mathbf{s}_{t-1}), \tag{1}$$

where g is usually a deterministic non-linear transition function. g gains extra strength to capture long-term memories when implemented with gated activation functions [6] such as long short-term memory [LSTM, 7] or gated recurrent unit [GRU, 5]. We can add more hidden layers in advance or subsequent to the RNN to increase the capacity of the model such that:

$$\mathbf{z}_t = h(g(f(\mathbf{x}_t), \mathbf{s}_{t-1})), \tag{2}$$

where, a sequence $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)$ is the transformed feature representation of the input sequence $\mathbf{x}$, and f and h are additional hidden layers. Instead of using the whole $\mathbf{z}$, we can apply a context function c to reduce the dimensionality and take only the abstract information out of $\mathbf{z}$. The context function c can be either defined as introduced in [5]:

$$c_1(\mathbf{z}) = \mathbf{z}_T, \tag{3}$$

or as introduced in [1]:

$$c_2(\mathbf{z}) = \sum_{t=1}^{T} \alpha_t \mathbf{s}_t, \tag{4}$$

where α_t is the weight of each annotation $\mathbf{h}_t$. $c(\mathbf{z})$ can be used as the learned features for the logistic regression classifier:

$$\hat{y} = \sigma(c(\mathbf{z})), \tag{5}$$

where σ is a notation of a sigmoid function.

In this study, we implement f and h with deep neural networks (DNNs) using fully-connected layers and rectified linear units [11] as non-linearity, g with a recurrent layer using either GRU or LSTM as the state transition function, and the context function c with either Eq. (3) or Eq. (4).

3 Proposed Models

We take a neural network based approach where we can stack multiple feedforward and recurrent layers to learn hierarchical features from the training examples and the objective function via stochastic gradient descent.

We consider two types of inputs, which are text transcripts and audio signals of utterances. Depending on what types of inputs are used, we can divide the models into three groups: (1) receive only text inputs, (2) receive only audio inputs and (3) receive both inputs. When a model receives both inputs as (3), we can think of a simple but naive way of combining two different features as shown as 'Combinational' in Fig. 1. For a model in each group, it can choose the context function to become as either Eq. (3) or Eq. (4), so the number of combinations becomes six. However, there is another model, receives both inputs, uses Eq. (4) as the context function, but uses a different way of combining two features that is depicted as 'Conditional' in Fig. 1.
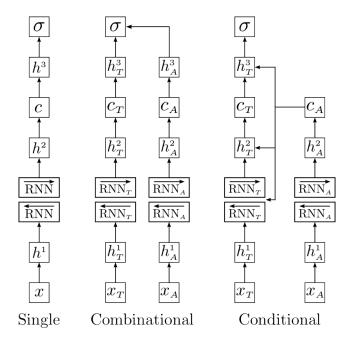

Figure 1: Graphical illustration of each group, note that the context function c can be implemented as either Eq. (3) or Eq. (4). x of 'Single' model can be either a text input or an audio input. 'Combinational' and 'Conditional' models take both inputs, where the subscript $_T$ stands for text source and $_A$ stands for audio source. The topmost blocks with σ indicate the logistic regression classifiers. Each layer has 200 hidden units. The training objective is to minimize the average negative log-likelihood of the training examples. We implement the RNN with its bidirectional variant [12].

For each model in each group, we train it with three different ways: (1) without any regularization methods, (2) use dropout [13] and (3) use batch normalization (BN) [8] (note that we are not the first to apply batch normalization to a neural network architecture that contains an RNN [9]). However, there is another diversity, the state transition function of the RNN hidden state, which can be implemented either as a GRU or an LSTM. Therefore, for each model, there are six different candidates to compare with. Recall that we have seven different models, each model has six different variations, there is a total of 42 candidates to be tested on two datasets that are MSR-Skype and CALLHOME.

4 Experiment Settings

4.1 Datasets

MSR-Skype   MSR-Skype dataset contains $18,006$ examples given as text-audio pairs, and the proportion of positive and negative examples are well-balanced. Each example is an utterance, which is segmented manually. We only use examples that contain 3 to 25 words to train the models. We use 80% of the examples as a training set and reserve 20% of the examples to validate and evaluate the models.

Table 2: Test F1 score of the models trained on MSR-Skype. First two columns use neither dropout (shortened as D) nor BN.

	GRU	LSTM	GRU, D	LSTM, D	GRU, BN	LSTM, BN
text, c_1	88.8	88.6	89.1	89.1	90.6	90.2
text, c_2	89.5	88.9	88.9	88.7	90.8	90.5
audio, c_1	77.3	73.7	79.2	77.0	75.3	71.2
audio, c_2	77.2	77.6	80.7	81.2	76.5	76.8
combination, c_1	89.2	88.9	89.1	88.8	90.5	90.3
combination, c_2	89.7	88.9	89.2	89.0	90.9	91.0
condition, c_2	90.0	89.8	90.0	90.1	90.7	90.1

CALLHOME   We use a subset of the original CALLHOME, where the text transcripts are created by human annotators. There are $2,528$ examples given as text-audio pairs. Utterances are segmented manually, and the train/validation/test splits are divided as same as the MSR-Skype dataset.

Preprocessing   For the text data, we remove punctuations, commas, question marks, exclamation marks to prevent the model from making decisions based on these special tokens. We do not consider pretraining word representation vectors with external datasets, however, they are learned jointly with the objective function during the training procedure. Therefore, h^1 in 'Single' (only when x is text data) and h^1_τ in 'Combinational' and 'Conditional' become continuous vector representations of the words (in this case, we do not apply non-linearity). We built the dictionary from MSR-Skype and CALLHOME, which contains 13,911 vocabularies.

We extract MFCC from the raw audio signals with $40ms$ frame duration, and $15ms$ overlap. The lengths of the audio sequences (after extracting MFCC) could be significantly longer than the text sequences, therefore, in order to reduce the number of timesteps, we concatenate four frames into one chunk and treat it as a single frame.

4.2   Results

Table 2 shows the results of the models trained on MSR-Skype dataset. We can observe a few tendencies in the obtained results depending on what kind of variations are applied to the models (c_1 or c_2, GRU or LSTM, dropout or batch normalization and types of inputs).

In general, using both input sources are helpful, but the advantage is not that impressive when batch normalization is used for training. The lengths of the audio sequences are usually longer than the text sequences, and attention mechanism (c_2) [1] is known to be a nice solution to deal with long sequences. Therefore, when the model can only take audio inputs, c_2 is a better option than c_1.

Dropout will help in most cases, however, when using both input sources, the performance does not improve that much. In fact, the performance gets worse than the models, which do not use dropout. We assume that the optimization problem becomes difficult with dropout when the models receive both input sources, hence, in this case we need more care in using dropout. Batch normalization improves the performance with a huge gap for the models that receive text source as inputs. However, batch normalization does not help the models that can only receive audio inputs. The best performance is achieved by a model that receives both input sources (combinational), uses c_2 as context function, uses batch normalization for training and uses LSTM as the state transition function of the RNN.

Table 3 shows the result of each model trained on CALLHOME. We can observe that c_2 helps the models that only take audio inputs, and batch normalization improves the performance of the models that includes text source as their inputs. The best performance is achieved by a model that takes both input sources (combinational), uses c_1 as context function, uses batch normalization for training and uses GRU as the state transition function of the RNN.

In Table 4, we test our models on sequences with different lengths. We use the same models that were trained on MSR-Skype, without any regularization methods. The sequences are divided into

Table 3: Test F1 score of the models trained on CALLHOME. First two columns use neither dropout (shortened as D) nor BN.

	GRU	LSTM	GRU, D	LSTM, D	GRU, BN	LSTM, BN
text, c_1	81.3	81.1	81.4	80.6	82.9	82.7
text, c_2	81.0	80.7	81.2	81.5	83.5	83.0
audio, c_1	70.9	70.0	72.2	72.4	67.0	66.3
audio, c_2	70.8	71.8	72.5	73.4	69.2	67.8
combination, c_1	83.1	82.7	82.6	82.8	84.6	84.0
combination, c_2	83.0	82.5	82.7	82.4	84.1	83.6
condition, c_2	83.8	83.9	83.9	83.9	82.4	83.7

Table 4: Test F1 score of the models trained on MSR-Skype and tested on variable-length sequences.

	text, c_2	audio, c_2	combination, c_2	condition, c_2
Short Sequences	77.6	64.6	78.3	79.3
Intermediate Sequences	90.1	82.3	90.5	91.2
Long Sequences	80.3	69.4	82.8	84.0

three groups depending on the number of words contained in each sequence. Short sequences have less than 5 words, long sequences have more than 20 words, and intermediate sequences contain 5 to 20 words. We observe that the models achieve the best performance on intermediate sequences, and the models tend to do better jobs on short and long sequences when the inputs contain text source. The performance degradations on short or long sequences compared to intermediate sequences are smaller when we use both input sources (see 'combination' and 'condition', especially models lose less performance against long sequences).

Table 5 shows some test examples that neither contain *wh*-words nor have canonical form of questions, which we have already introduced as declarative questions in Sec. 2.1. In this kind of questions, there are usually rising pitches at the end of the audio signals. For the models, which receive the audio source as inputs, can benefit from having audio information as shown in Table 5 (see 'audio', 'combination' and 'condition'). For the models, which receive only text source as inputs, do not have relevant information to guess whether the given utterances are questions or not.

The predicted scores from the models using both inputs are sometimes less than the scores from the model using only audio inputs. We assume that these models have to make compromise between the text features and audio features when these two are in conflict. However, given the training objective, it is difficult to expect that the models will completely ignore one of the features, instead, the models will tend to learn more smooth decision boundaries.

5   Conclusion

We explore various types of RNN-based architectures for detecting questions in English utterances. We discover some features that can help the models to achieve better scores in the question detection task. Different types of inputs can complement each other, and the models can benefit from using both text and audio sources as inputs. Attention mechanism (c_2) helps the models that receive long audio sequences as inputs. Regularization methods can help the models to generalize better, however, when the models receive multimodal inputs, we need to be more careful on using these regularization methods.

Acknowledgments

Table 5: Examples of predicted scores on declarative questions.

Test Examples	text, c_2	audio, c_2	combination, c_2	condition, c_2
any other questions?	0.44	0.98	0.72	0.84
and your cats?	0.63	0.93	0.97	0.72
oh, the bird?	0.42	0.83	0.77	0.72

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] T. Bazillon, B. Maza, M. Rouvier, F. Bechet, and A. Nasr. Speaker role recognition using question detection and characterization. In *INTERSPEECH*, pages 1333–1336, 2011.

[4] K. Boakye, B. Favre, and D. Hakkani-Tür. Any questions? automatic question detection in meetings. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 485–489. IEEE, 2009.

[5] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

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

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, 2015.

[9] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio. Batch normalized recurrent neural networks. *arXiv preprint arXiv:1510.01378*, 2015.

[10] D. Metzler and W. B. Croft. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504, 2005.

[11] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[12] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] K. Wang and T.-S. Chua. Exploiting salient patterns for question detection and question retrieval in community-based question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1155–1163. Association for Computational Linguistics, 2010.

[15] J. Yuan and D. Jurafsky. Detection of questions in chinese conversational speech. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 47–52. IEEE, 2005.