A Nil-Aware Answer Extraction Framework for Question Answering

Souvik Kundu and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
{souvik, nght}@comp.nus.edu.sg

Abstract

Recently, there has been a surge of interest in reading comprehension-based (RC) question answering (QA). However, current approaches suffer from an impractical assumption that every question has a valid answer in the associated passage. A practical QA system must possess the ability to determine whether a valid answer exists in a given text passage. In this paper, we focus on developing QA systems that can extract an answer for a question if and only if the associated passage contains an answer. If the associated passage does not contain any valid answer, the QA system will correctly return *Nil*. We propose a nil-aware answer span extraction framework that is capable of returning *Nil* or a text span from the associated passage as an answer in a single step. We show that our proposed framework can be easily integrated with several recently proposed QA models developed for reading comprehension and can be trained in an end-to-end fashion. Our proposed nil-aware answer extraction neural network decomposes pieces of evidence into relevant and irrelevant parts and then combines them to infer the existence of any answer. Experiments on the NewsQA dataset show that the integration of our proposed framework significantly outperforms several strong baseline systems that use pipeline or threshold-based approaches.

1 Introduction

Machine comprehension (MC) systems mimic the process of reading comprehension (RC) by answering questions after understanding natural language text. Several datasets and resources have been developed recently. Richardson et al. (2013) developed a small-scale multiple-choice question answering (QA) dataset. Hermann et al. (2015) created a large cloze-style MC dataset based on CNN and Daily Mail news article summaries. However, Chen et al. (2016) reported that the task is not challenging enough and hence, advanced models had to be evaluated on more realistic datasets. Subsequently, SQuAD (Rajpurkar et al., 2016) was released, where, unlike previous datasets, the answers to different questions can vary in length.

In previous datasets, questions and answers are formulated given text passages. Hence, a valid answer can always be found in the associated passage for every question created. Trischler et al. (2017) proposed a more challenging and realistic dataset, NewsQA, where the questions were formed using CNN article summaries without accessing the original full texts. As such, some questions have no valid answers in the associated passages (referred to as nil questions).

Recently, several neural models for answer span extraction have been proposed (Wang and Jiang, 2017; Seo et al., 2017; Yang et al., 2017; Xiong et al., 2017; Weissenborn et al., 2017; Wang et al., 2017; Shen et al., 2017b; Chen et al., 2017; Kundu and Ng, 2018). However, none of the models considered nil questions, although it is crucial for a practical QA system to be able to determine whether a text passage contains a valid answer for a question. In this paper, we focus on developing QA systems that extract an answer for a question if and only if the associated passage contains a valid answer. Otherwise, they are expected to return *Nil* as answer. We propose a nil-aware answer extraction framework which returns *Nil* or a span of text as answer, when integrated with end-to-end neural MC models. Our proposed framework is based on evidence decomposition-aggregation, where the evidence vectors derived by a higher level encoding layer are first decomposed into relevant and irrelevant components and later aggregated to infer the existence of a valid answer. In addition, we develop several baseline models with pipeline and threshold-based approaches. In a pipeline model, detection of nil questions is carried out separately before answer span extraction. In a threshold-based model, the answer span extraction model is entirely trained on questions that have valid answers, and *Nil* is returned based on a confidence threshold.

The contributions of this paper are as follows: (1) We propose a nil-aware answer span extraction framework to return *Nil* or an exact answer span to a question, in a single step, depending on the existence of a valid answer. (2) Our proposed framework can be readily integrated with many recently proposed neural machine comprehension models. In this paper, we extend four machine comprehension models, namely BiDAF (Seo et al., 2017), R-Net (Wang et al., 2017), DrQA (Chen et al., 2017), and AMANDA (Kundu and Ng, 2018), with our proposed framework, and show that they achieve significantly better results compared to the corresponding pipeline and threshold-based models on the NewsQA dataset.

2 Task Definition

Given a passage and a question, we propose models that can extract an answer if and only if the passage contains an answer. When the passage does not contain any answer, the models return *Nil* as the answer. A valid answer is denoted as two pointers in the passage, representing the start and end tokens of the answer span. Let $\mathcal{P}$ be a passage with tokens $(\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_T)$ and $\mathcal{Q}$ be a question with tokens $(\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_U)$, where T and U are the length of the passage and question respectively. A system needs to determine whether the answer is *Nil* or comprises two pointers, b and e, such that $1 \leq b \leq e \leq T$.

3 Proposed Framework

In this section, we first describe our proposed evidence decomposition-aggregation framework for nil-aware answer extraction. Then, we provide a detailed description of how we extend a state-of-the-art model AMANDA (Kundu and Ng, 2018) to NAMANDA[1] (nil-aware AMANDA). We also provide brief descriptions of how we integrate our proposed framework with the other three models.

[1]Our source code is available at `https://github.com/nusnlp/namanda`

3.1 Nil-Aware Answer Extraction

Decomposition of lexical semantics over sentences has been successfully used in the past for sentence similarity learning (Wang et al., 2016). Most of the recently proposed machine reading comprehension models can be generalized based on a common pattern observed in their network architecture. They have a question-passage joint encoding layer (also known as question-aware passage encoding layer) followed by an evidence encoding layer. In this work, we decompose the evidence vectors for each passage word obtained from the evidence encoding layer with respect to question-passage joint encoding vectors to derive semantically relevant and irrelevant components. We decompose the evidence vectors for each passage word, because passage vectors can be partially supported by the corresponding question-passage joint encoding vectors, and based on the level of support, it either increases or decreases the chance of finding a valid answer. When we aggregate the orthogonally decomposed evidence vectors, it combines both the supportive and unsupportive pieces of evidence for a particular passage word. To obtain the most impactful portions, we perform a max-pooling operation over all the aggregated vectors. The resulting vector is denoted as the Nil vector. As the training set contains both nil questions (with no valid answers) and non-nil questions (with valid answers), the model automatically learns when to pool unsupportive (for nil questions) and supportive (for non-nil questions) portions to construct the Nil vector. In this way, the model is able to induce a strong bias towards the nil pointer when there is no answer present due to the dominance of unsupportive components in the nil vector.

The proposed method in Wang et al. (2016) was developed for sentence similarity learning tasks, such as answer sentence selection. They decompose an answer sentence with respect to a question and vice versa. The decomposed vectors are then aggregated to obtain a single vector which is used to derive the similarity score. Although our proposed method (developed for the more complex task of answer span extraction) is inspired from the idea of lexical decomposition and composition, one major difference is that we decompose the evidence vectors with respect to question-passage joint encoding vectors. Another important advance is how it is adopted to return nil or a span

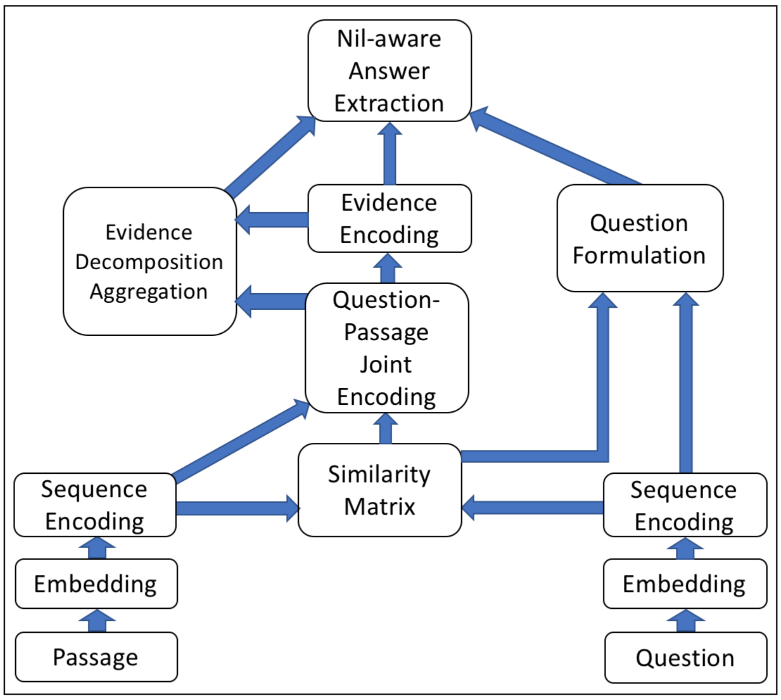


Figure 1: Overview of the architecture of Nil-aware AMANDA (NAMANDA).

of text from the passage in a single step.

3.2 Nil-Aware AMANDA

The architecture of Nil-aware AMANDA (NAMANDA) is given in Figure 1.

3.2.1 Embeddings

To obtain the embeddings, we concatenate word and character-level embedding vectors. We use pre-trained vectors from GloVe (Pennington et al., 2014) for word-level embeddings. For character embeddings, a trainable character-based lookup table is used followed by a convolutional neural network (CNN) and max-pooling (Kim, 2014).

3.2.2 Sequence Encoding

We use bi-directional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) on the embedding vectors to incorporate contextual information. We represent the outputs as $\mathbf{D} \in \mathbb{R}^{T \times H}$ and $\mathbf{Q} \in \mathbb{R}^{U \times H}$ for passage and question respectively. H is the number of hidden units of the BiLSTMs.

3.2.3 Similarity Matrix

The similarity matrix is obtained by computing the dot product of passage and question sequence-level encoding vectors. The similarity matrix $\mathbf{A} \in \mathbb{R}^{T \times U}$ can be expressed as $\mathbf{A} = \mathbf{D}\ \mathbf{Q}^\top$, where $A_{i,j}$ is the similarity between the ith passage word and the jth question word.

3.2.4 Question Formulation

To aggregate the most relevant parts of the question, column-wise maximum values of $\mathbf{A}$ are normalized using a softmax function to obtain $\mathbf{k} \in \mathbb{R}^U$. Then, the question vectors in $\mathbf{Q}$ are aggregated by $\mathbf{q}_{ma} = \mathbf{k}\ \mathbf{Q}$. The question type information is incorporated via $\mathbf{q}_f \in \mathbb{R}^{2H}$, by concatenating the sequence-level question encoding vectors of the first wh-word $\mathbf{q}_{t_{wh}}$ and its following word $\mathbf{q}_{t_{wh}+1}$. It can be given as $\mathbf{q}_f = \mathbf{q}_{t_{wh}} \,||\, \mathbf{q}_{t_{wh}+1}$, where $||$ denotes the concatenation operation. The set of wh-words we used is $\{$*what, who, how, when, which, where, why*$\}$. The final question representation, $\tilde{\mathbf{q}} \in \mathbb{R}^H$, is formulated by applying a feed-forward neural network on the concatenated representation of $\mathbf{q}_{ma}$ and $\mathbf{q}_f$.

3.2.5 Question-Passage Joint Encoding

In this step, we jointly encode the passage and question. We apply a row-wise softmax function on $\mathbf{A}$ to obtain $\mathbf{R} \in \mathbb{R}^{T \times U}$. Now, for all the passage words, the aggregated question representation $\mathbf{G} \in \mathbb{R}^{T \times H}$ is computed by $\mathbf{G} = \mathbf{R}\ \mathbf{Q}$. The aggregated question vectors corresponding to the passage words are then concatenated with the sequence-level passage vectors to obtain $\mathbf{S} \in \mathbb{R}^{T \times 2H}$. We apply another BiLSTM to obtain a combined representation $\mathbf{V} \in \mathbb{R}^{T \times H}$.

3.2.6 Evidence Decomposition-Aggregation

First, multi-factor self-attentive encoding is applied to accumulate evidence from the entire passage. The use of multiple factors while calculating self attention helps to obtain meaningful information from a long context with fine-grained inference. If m represents the number of factors, multi-factor attention $\mathbf{F}^{[1:m]} \in \mathbb{R}^{T\times m\times T}$ is formulated as:

$$\mathbf{F}^{[1:m]} = \mathbf{V}\,\mathbf{W}_f^{[1:m]}\,\mathbf{V}^\top , \tag{1}$$

where $\mathbf{W}_f^{[1:m]} \in \mathbb{R}^{H\times m\times H}$ is a 3-way tensor. Now, to refine the evidence, a max-pooling operation is performed on $\mathbf{F}^{[1:m]}$ over the number of factors, resulting in the self-attention matrix $\mathbf{F} \in \mathbb{R}^{T\times T}$. We normalize $\mathbf{F}$ by applying a row-wise softmax function, resulting in $\tilde{\mathbf{F}} \in \mathbb{R}^{T\times T}$. Now the self-attentive encoding $\mathbf{M} \in \mathbb{R}^{T\times H}$ can be given as $\mathbf{M} = \tilde{\mathbf{F}}\,\mathbf{V}$. The self-attentive encoding vectors are then concatenated with the question-dependent passage word encoding vectors ($\mathbf{V}$), and a feed-forward neural network-based gating is applied to control the overall impact, resulting in $\mathbf{Y} \in \mathbb{R}^{T\times 2H}$.

Then we decompose the evidence vector for every passage word with orthogonal decomposition. Each row of $\mathbf{Y}$, $\mathbf{y}_t \in \mathbb{R}^{2H}$, is decomposed into its parallel and perpendicular components with respect to the corresponding question-passage joint encoding ($\mathbf{S}$) vector, $\mathbf{s}_t \in \mathbb{R}^{2H}$. The parallel components represent the relevant parts of the accumulated evidence and the orthogonal components represent the irrelevant counterparts. If the parallel component of $\mathbf{y}_t$ is represented as $\mathbf{y}_t^{=} \in \mathbb{R}^{2H}$ and the perpendicular component is represented as $\mathbf{y}_t^{\perp} \in \mathbb{R}^{2H}$, then

$$\mathbf{y}_t^{=} = \frac{\mathbf{y}_t\,\mathbf{s}_t^\top}{\mathbf{s}_t\,\mathbf{s}_t^\top}\,\mathbf{s}_t \tag{2}$$

$$\mathbf{y}_t^{\perp} = \mathbf{y}_t - \mathbf{y}_t^{=} \tag{3}$$

Similarly, we derive the parallel and orthogonal vectors for all the passage words. We denote parallel components with $\mathbf{Y}^{=} \in \mathbb{R}^{T\times 2H}$ and perpendicular components with $\mathbf{Y}^{\perp} \in \mathbb{R}^{T\times 2H}$.

In the aggregation step, the parallel and orthogonal components are fed to a feed-forward linear layer. $\mathbf{Y}^a \in \mathbb{R}^{T\times H}$ denotes the output of the linear layer and $\mathbf{y}_t^a \in \mathbb{R}^H$ is its tth row:

$$\mathbf{y}_t^a = \tanh(\mathbf{y}_t^{=}\mathbf{W}_a + \mathbf{y}_t^{\perp}\mathbf{W}_a + \mathbf{b}_a) , \tag{4}$$

where $\mathbf{W}_a \in \mathbb{R}^{2H\times H}$ and $\mathbf{b}_a \in \mathbb{R}^H$ are the weight matrix and bias vector respectively. Then we apply a max-pooling operation over all the words to obtain the *Nil* vector representation denoted as $\hat{\mathbf{n}}$. Now we derive the score for the *Nil* pointer which will be shared for normalizing the beginning and ending pointers later. The *Nil* pointer score is given as:

$$n_s = \hat{\mathbf{n}}\mathbf{w}_n^\top , \tag{5}$$

where $\mathbf{w}_n \in \mathbb{R}^H$ is a learnable weight vector.

3.2.7 Nil-Aware Pointing

Two stacked BiLSTMs are used on top of $\mathbf{Y}$ to determine the beginning and ending pointers. Let the hidden unit representations of these two BiLSTMs be $\mathbf{B} \in \mathbb{R}^{T\times H}$ and $\mathbf{E} \in \mathbb{R}^{T\times H}$. We measure the similarity scores between the previously derived question vector $\tilde{\mathbf{q}}$ and the contextual encoding vectors in $\mathbf{B}$ and $\mathbf{E}$. If $\mathbf{s}_b \in \mathbb{R}^T$ and $\mathbf{s}_e \in \mathbb{R}^T$ are the scores for the beginning and ending pointers, then

$$\mathbf{s}_b = \tilde{\mathbf{q}}\,\mathbf{B}^\top , \quad \mathbf{s}_e = \tilde{\mathbf{q}}\,\mathbf{E}^\top \tag{6}$$

We prepend the *nil* score n_s to $\mathbf{s}_b$ and $\mathbf{s}_e$ for shared normalization. The updated scores $\hat{\mathbf{s}}_b \in \mathbb{R}^{T+1}$ and $\hat{\mathbf{s}}_e \in \mathbb{R}^{T+1}$ can be represented as:

$$\hat{\mathbf{s}}_b = [n_s, \mathbf{s}_b] , \quad \hat{\mathbf{s}}_e = [n_s, \mathbf{s}_e] \tag{7}$$

The beginning and ending pointer probability distributions for a given passage $\mathcal{P}$ and a question $\mathcal{Q}$ is given as:

$$\begin{aligned} \Pr(b \mid \mathcal{P}, \mathcal{Q}) &= \text{softmax}(\hat{\mathbf{s}}_b) \\ \Pr(e \mid \mathcal{P}, \mathcal{Q}) &= \text{softmax}(\hat{\mathbf{s}}_e) \end{aligned} \tag{8}$$

The joint probability distribution for answer a is given as:

$$\Pr(a \mid \mathcal{P}, \mathcal{Q}) = \Pr(b \mid \mathcal{P}, \mathcal{Q})\,\Pr(e \mid \mathcal{P}, \mathcal{Q}) \tag{9}$$

For training, we minimize the cross entropy loss summing over all training instances. During prediction, we select the locations in the passage for which the product of $\Pr(b)$ and $\Pr(e)$ is maximized, where $1 \le b \le e \le T+1$. If the value of b is 1, we assign the answer as *Nil*.

3.3 Nil-Aware DrQA

We extend DrQA (Chen et al., 2017) to NDrQA by integrating our proposed nil-aware answer extraction framework. In DrQA, the embeddings of passage tokens consist of pretrained word vectors from Glove, several syntactic features, and passage-question joint embedding (aligned question embedding). The syntactic features include exact match of passage words with question in surface, lowercase, and lemma form. They also used part-of-speech tags, named entity tags, and term frequency values for each passage word. Subsequently, a stack of BiLSTMs is used for encoding. The outputs of the stacked BilSTMs are used as evidence vectors to help extract the answer span. We decompose those stacked BiLSTM output vectors with respect to the passage embedding and generate the *nil* pointer score as given in Eqs (2–5). The question vector formulation in DrQA is performed by applying a stack of BilSTMs on question embedding. The nil-aware pointing mechanism is the same as that given in Section 3.2.7 except an additional bi-linear term is used for each $\mathbf{s}_b$ and $\mathbf{s}_e$ in Eq (6).

3.4 Nil-Aware R-Net

In R-Net (Wang et al., 2017), after embedding and encoding of the passage and question words, a gated recurrent network is used to obtain the question-passage joint representation. Subsequently, a self-matching attentive encoding is used to accumulate evidence from the entire passage. In the output layer, an answer recurrent pointer network is used to predict the boundary of an answer span. To extend R-Net to nil-aware R-Net (NR-Net), we decompose the output vectors of the self-matching layer with respect to the question-passage joint encoding vectors, and then aggregate them to obtain the *nil* pointer score as illustrated in Eqs (2–5). In the output layer, we combine the *nil* pointer score to the beginning and ending pointer unnormalized scores, and jointly normalize them using softmax function as given in Eqs (7–8).

3.5 Nil-Aware BiDAF

In BiDAF (Seo et al., 2017), an attention flow layer is used to jointly encode the passage and question. Then, a modeling layer is used to capture the interaction among the question-aware passage vectors. The output of the modeling layer serves as evidence to help extract the answer span in the output layer. To extend the BiDAF model to nil-aware BiDAF (NBiDAF), we decompose the output of the modeling layer with respect to the question-passage joint encoding vectors, and then aggregate them to derive the *nil* pointer score (similar to Eqs (2–5). Similar to the other nil-aware models, we concatenate the *nil* pointer score to the start and end pointer unnormalized scores derived in the output layer, and then jointly normalize them.

4 Baseline Models

For comparison, we propose two types of baseline approaches for nil-aware answer extraction.

4.1 Pipeline Approach

Here, two models are used in a pipeline:

Nil detector: Given a pair of passage and question, a nil detector model determines whether a valid answer is present in the passage.

Answer span extractor: If the nil detector model predicts the presence of a valid answer, the answer span extractor then extracts the answer.

For nil detection, we developed a logistic regression (LR) model with manually defined features and four neural models. For the LR model, we extract four different features which capture the similarity between a passage and a question. Let $\mathcal{P}$ be the passage and $\mathcal{Q}$ be the question (consisting of U' tokens excluding stop words). If $f(\mathcal{P},\ \mathcal{Q}_i)$ is the frequency of the ith question word in passage $\mathcal{P}$, then the first feature η is defined as:

$$\eta = \sum_{i=1}^{U'} \log(1 + f(\mathcal{P}, \mathcal{Q}_i)) \tag{10}$$

The second feature is the same as η, except that the lemma form is considered for both passage and question tokens instead of the surface form. Additionally, we include word overlap count features in both surface and lemma forms.

We also develop several advanced neural network architectures for nil detection. After embedding (the same as Section 3.2.1), we apply sequence-level encoding with either BiLSTM or CNN. For CNN, we use equal numbers of unigram, bigram, and trigram filters and the outputs are concatenated to obtain the final encoding. Next, we apply either global max-pooling (MP) or attentive pooling (AP) over all the sequence vectors to obtain an aggregated vector representation. Let the sequence encoding of a passage be $\mathbf{P}^{nd} \in \mathbb{R}^{T \times H}$, and $\mathbf{p}_t^{nd}$ be the tth row of $\mathbf{P}^{nd}$. The aggregated vector $\tilde{\mathbf{p}}_{nd} \in \mathbb{R}^H$ for AP can be obtained as:

$$a_t^{nd} \propto \exp(\mathbf{p}_t^{nd} \mathbf{w}^\top) \quad (11)$$
$$\tilde{\mathbf{p}}_{nd} = \mathbf{a}^{nd} \mathbf{P}^{nd} \;, \quad (12)$$

where $\mathbf{w} \in \mathbb{R}^H$ is a learnable vector. Similarly, we derive the aggregated question vector $\tilde{\mathbf{q}}_{nd}$. For nil detection, we compute the similarity score (s_{nd}) between the aggregated vectors:

$$s_{nd} = \text{sigmoid}(\tilde{\mathbf{p}}_{nd}\; \tilde{\mathbf{q}}_{nd}^\top) \quad (13)$$

We experimented with four state-of-the-art answer span extractor models, namely BiDAF (Seo et al., 2017), R-Net (Wang et al., 2017), DrQA (Chen et al., 2017), and AMANDA (Kundu and Ng, 2018). Note that the answer extraction models are trained entirely on passage-question pairs which always have valid answers.

4.2 Threshold-Based Approach

Here, we do not use any nil questions to train the neural answer span extraction model. This approach assumes that when there is a valid answer, the probability distributions of the beginning and ending pointers will have lower entropy. This results in a higher maximum joint probability of the beginning and ending pointers. In contrast, when an answer is not present in the associated passage, the output probability distributions have higher entropy, resulting in a lower value of maximum joint probability. We set the maximum joint probability *threshold* based on the best Nil F1 score on the nil questions in the development set. Now, for a given test passage and question, we first compute the maximum of all the joint probabilities associated with all the answer spans. Let a_{span} be the answer span with highest joint probability p_{max}. We assign the final *answer* as follows:

$$answer = \begin{cases} Nil, & \text{if } p_{max} \leq threshold \\ a_{span}, & \text{otherwise} \end{cases} \quad (14)$$

5 Experiments

5.1 Experimental Settings

We use the NewsQA dataset with nil questions (Trischler et al., 2017) in our experiments. Its training, development, and test sets consist of 10,938, 638, and 632 passages respectively and every passage is associated with some questions. In each subset, there are some questions which

Dataset		#Passages	#Questions
Train	NewsQA	10,938	92,549
	+Nil Qs		107,673
Dev	NewsQA	638	5,166
	+Nil Qs		5,988
Test	NewsQA	632	5,126
	+Nil Qs		5,971

Table 1: Statistics of the NewsQA dataset.

have no answers in the corresponding associated passages (i.e., the nil questions). The detailed statistics of the dataset are given in Table 1.

We compute exact match (EM) and F1 score for questions with valid answers. For questions without any valid answers, we compute Nil precision, recall, and F1 scores as follows:

$$\text{Nil precision} = \frac{\#\text{Correctly predicted Nil}}{\#\text{predicted Nil}} \quad (15)$$
$$\text{Nil recall} = \frac{\#\text{Correctly predicted Nil}}{\#\text{Nil questions}} \quad (16)$$
$$\text{Nil F1} = 2 \times \frac{\text{Nil precision} \times \text{Nil recall}}{\text{Nil precision} + \text{Nil recall}} \quad (17)$$

To compute the overall EM and F1 scores, we consider *Nil* as correct for the questions which do not have any valid answers. All evaluation scores reported in this paper are in %.

All the neural network models are implemented in PyTorch[2]. We use the default hyper-parameters for all the answer span extractor models. We use the open source implementation of DrQA[3]. We use a third party implementation of R-Net[4] whose performance is very similar to the original scores. We reimplemented BiDAF[5] and AMANDA[6] to easily integrate our proposed nil-aware answer extraction framework and make the training faster. We integrate the nil-aware answer span extraction framework with each model keeping all the hyper-parameters unchanged. For nil-detection models, we use the same settings as (N)AMANDA. We use 300 hidden units for BiLSTMs and a total of 300 filters for the CNN-based models. We use dropout (Srivastava et al., 2014) with probability 0.3 for every trainable layer. We use binary cross-entropy loss and the Adam optimizer (Kingma and Ba, 2015) for training the nil-detection models.

[2]http://pytorch.org

[3]https://github.com/facebookresearch/DrQA

[4]`https://github.com/HKUST-KnowComp/MnemonicReader/blob/master/r_net.py`

[5]Our implementation gives 3% lower F1 score compared to the reported results in Seo et al. (2017) on the SQuAD development set.

[6]Our implementation gives 0.5% higher F1 score compared to the reported scores in Kundu and Ng (2018) on the NewsQA test set.

Answer Extractor	Nil Detector	Test Set (w/o Nil Questions)		Test Set (with Nil Questions)				
		EM	F1	Nil Precision	Nil Recall	Nil F1	Overall EM	Overall F1
BiDAF	-	42.5	57.5	-	-	-	36.5	49.4
	LR	39.6	53.2	33.1	28.9	30.9	38.1	49.7
	MP-BiLSTM	40.1	54.2	52.5	48.3	50.3	41.3	53.4
	MP-CNN	42.3	57.2	73.8	15.0	24.9	38.4	51.2
	AP-BiLSTM	40.5	54.7	55.3	47.5	51.1	41.5	53.7
	AP-CNN	40.1	54.3	50.8	39.9	44.7	40.1	52.3
NBiDAF		40.8	54.7	48.0	59.6	53.2	43.5	55.4
R-Net	-	49.9	64.0	-	-	-	42.8	54.8
	LR	46.4	58.8	33.1	28.9	30.9	43.9	54.6
	MP-BiLSTM	47.3	60.3	52.5	48.3	50.3	47.5	58.6
	MP-CNN	49.7	63.6	73.8	15.0	24.9	44.8	56.7
	AP-BiLSTM	47.6	60.7	55.3	47.5	51.1	47.6	58.8
	AP-CNN	47.2	60.4	50.8	39.9	44.7	46.2	57.5
NR-Net		47.0	60.8	53.6	57.6	55.5	48.5	60.3
DrQA	-	50.0	64.0	-	-	-	42.9	54.8
	LR	46.3	58.8	33.1	28.9	30.9	43.8	54.6
	MP-BiLSTM	47.1	60.2	52.5	48.3	50.3	47.3	58.5
	MP-CNN	49.6	63.5	73.8	15.0	24.9	44.7	56.7
	AP-BiLSTM	47.4	60.6	55.3	47.5	51.1	47.4	58.8
	AP-CNN	47.0	60.2	50.8	39.9	44.7	46.0	57.3
NDrQA		48.5	61.8	53.5	57.2	55.3	49.8	61.1
AMANDA	-	49.2	64.2	-	-	-	42.2	55.1
	LR	45.4	58.8	33.1	28.9	30.9	43.1	54.5
	MP-BiLSTM	46.2	60.2	52.5	48.3	50.3	46.5	58.5
	MP-CNN	48.3	63.3	73.8	15.0	24.9	43.6	56.5
	AP-BiLSTM	46.3	60.6	55.3	47.5	51.1	46.5	58.7
	AP-CNN	45.9	60.0	50.8	39.9	44.7	45.1	57.1
NAMANDA		48.6	62.2	57.1	56.7	56.9	49.7	61.5

Table 2: Performance Comparison with pipeline approaches on the NewsQA test set.

5.2 Results

Tables 2 and 3 compare results of the nil-aware answer span extractor models with several pipeline and threshold-based models, respectively. We also include the results of four standalone answer span extraction models on the test set without nil questions. Table 2 shows that the end-to-end nil-aware models achieve the highest overall EM and F1 scores compared to all the corresponding pipeline systems. Note that the MP-BiLSTM nil detection model achieves higher Nil F1 scores compared to LR and MP-CNN. This is because BiLSTM is able to capture long-range contextual information to infer the existence of valid answers. Furthermore, AP-based models perform better compared to MP-based models as the attention mechanism used in AP-based models inherently identifies important contextual information. Due to this, the performance gap between AP-CNN and AP-BiLSTM is lower than the performance gap between MP-CNN and MP-BiLSTM. In addition to achieving higher Nil F1 score than the strong nil detection baseline systems, nil-aware models manage to achieve competitive scores compared to the corresponding standalone answer span extractors on the test set where there are no nil questions.

Table 3 shows that the nil-aware models outperform the corresponding threshold-based models. Note that all four answer span extraction models, when used in a threshold-based approach for nil detection, produce low Nil precision and relatively higher Nil recall. The low precision significantly degrades performance on the test set without nil questions. These models often return *Nil* since it is critical to find suitable values for the required *threshold*. This is because NewsQA passages are often very long and as a result, probability distributions with higher entropy for answer pointer selection lead to irregular maximum joint probability *threshold* values.

We perform statistical significance tests using paired t-test and bootstrap resampling. Performances of all the nil-aware models (in terms of overall EM and F1) are significantly better ($p < 0.01$) than the corresponding best pipeline models and threshold-based approaches.

Answer Extractor	Nil Answer Handling	Test Set (w/o Nil Questions)		Test Set (with Nil Questions)				
		EM	F1	Nil Precision	Nil Recall	Nil F1	Overall EM	Overall F1
BiDAF	No	42.5	57.5	-	-	-	36.5	49.4
	Yes	37.9	48.3	25.8	60.2	36.1	41.0	50.0
NBiDAF		40.8	54.7	48.0	59.6	53.2	43.5	55.4
R-Net	No	49.9	64.0	-	-	-	42.8	54.8
	Yes	45.3	54.7	25.5	53.6	34.6	46.5	54.5
NR-Net		47.0	60.8	53.6	57.6	55.5	48.5	60.3
DrQA	No	50.0	64.0	-	-	-	42.9	54.8
	Yes	40.8	48.1	23.1	68.0	34.5	44.6	51.0
NDrQA		48.5	61.8	53.5	57.2	55.3	49.8	61.1
AMANDA	No	49.2	64.2	-	-	-	42.2	55.1
	Yes	42.2	51.3	24.0	59.5	34.2	44.6	52.5
NAMANDA		48.6	62.2	57.1	56.7	56.9	49.7	61.5

Table 3: Performance comparison with threshold-based approaches on the NewsQA test set.

5.3 Analysis

For better understanding, we present further experiments and analysis of one of the proposed models, NAMANDA.

In addition to linear aggregation, we experiment with BiLSTM-based and CNN-based aggregation models. When we use BiLSTM aggregation, Eq. (4) is modified to $\mathbf{y}_t^a = \mathbf{h}_t^= + \mathbf{h}_t^\perp$, where

$$\begin{aligned} \mathbf{h}_t^= &= \text{BiLSTM}(\mathbf{y}_t^=, \mathbf{h}_{t-1}^=, \mathbf{h}_{t+1}^=) \quad (18) \\ \mathbf{h}_t^\perp &= \text{BiLSTM}(\mathbf{y}_t^\perp, \mathbf{h}_{t-1}^\perp, \mathbf{h}_{t+1}^\perp) \end{aligned}$$

We use equal numbers of unigram, bigram, and trigram filters for CNN-based aggregation. Similar to BiLSTM-based aggregation, we add the CNN outputs for $\mathbf{Y}^=$ and $\mathbf{Y}^\perp$. Table 4 shows that linear aggregation achieves the highest overall F1 score despite using the least number of parameters.

Table 5 shows the results of NAMANDA on the NewsQA development set when different components are removed such as character embeddings, question-passage joint encoding, and the second LSTM for the answer-ending pointer. When question-passage joint encoding is removed, self-attentive encoding is formed as well as decomposed with respect to sequence-level passage encoding. When we remove the second LSTM for the answer-ending pointer, a feed-forward network is used instead. It is clear from Table 5 that question-passage joint encoding has the highest impact.

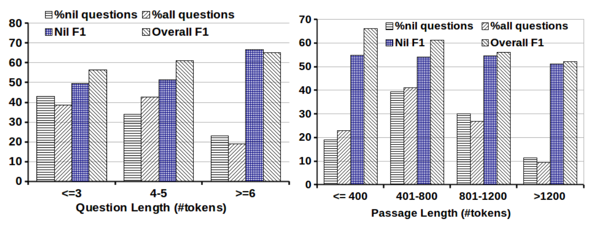
Figure 2(a) and Figure 2(b) show the results of NAMANDA on different question (excluding the stop words) and passage lengths respectively on the NewsQA development set. With increasing question length, the Nil F1 score also improves. This is because with more information in a question, it becomes easier to detect whether the associated passage contains a valid answer. Increasing Nil F1 scores also help to improve the overall F1 scores. However, the overall F1 score degrades with increasing length of the associated passage. When the associated passage is long, it is difficult for the answer span extractor to extract an answer for a question which has a valid answer, due to the increasing amount of potentially distracting information. The Nil F1 scores remain similar for passages consisting of not more than 1,200 tokens. Beyond that, the Nil F1 score degrades a little

Aggregation Model	w/o Nil	with Nil	
	EM (F1)	Nil Prec/ Rec (F1)	Overall EM (F1)
BiLSTM	47.6 (60.9)	56.5/53.5 (55.0)	48.4 (59.9)
CNN	46.2 (60.0)	52.7/54.5 (53.6)	47.3 (59.2)
Linear (NAMANDA)	47.8 (60.5)	51.2/57.2 (54.0)	49.1 (60.0)

Table 4: Effect of different aggregation models on the NewsQA dev set.

Model	w/o Nil	with Nil	
	EM (F1)	Nil Prec/ Rec (F1)	Overall EM (F1)
– character embeddings	46.3 (59.2)	51.9/54.1 (53.0)	47.4 (58.5)
– q-passage joint encoding	32.5 (43.9)	41.4/58.8 (48.6)	36.1 (45.9)
– second LSTM	47.6 (60.3)	56.7/51.0 (53.7)	48.0 (59.0)
NAMANDA	47.8 (60.5)	51.2/57.2 (54.0)	49.1 (60.0)

Table 5: Ablation studies on the NewsQA dev set.



(a) (b)

Figure 2: Results for different (a) question and (b) passage lengths on NewsQA dev set.

as it becomes very challenging to infer the existence of a valid answer accurately with increasing amount of potentially distracting information present in the passage.

Nil detection is itself a very challenging task. Performances of the nil-aware models are worse than the corresponding answer extractor models on the test set without nil questions as Nil precision is lower than 100%. We carried out an experiment to evaluate the performance of NAMANDA on development sets with varying number of nil questions. As the proportion of nil questions in a set increases, NAMANDA outperforms AMANDA by a larger margin on overall scores.

6 Related Work

In some years of the question answering track at the Text Retrieval Conference (TREC)[7], some questions were considered as nil questions for which no valid answers could be found in the entire corpus. Participating teams were required to return *Nil* as answer for those questions. Many teams used threshold-based methods to determine whether any of the retrieved answers for a given question was valid or not. If none of the answers had high confidence, *Nil* was returned as answer. To evaluate the performance on the nil questions, TREC used Nil precision, recall and F1 scores.

In recent years, research on question answering has witnessed substantial progress with rapid advances in neural network architectures. For example, on the answer sentence selection task, where a system has to choose the correct answer sentence from a pool of candidate sentences for a given question, the introduction of attention-based neural models has rapidly advanced the state of the art (Tan et al., 2015; Yang et al., 2016; dos Santos et al., 2016; Wang et al., 2016; Bian et al., 2017; Shen et al., 2017a).

However, in the answer sentence selection task, the answer is always a full sentence. Rajpurkar et al. (2016) released a reading comprehension-based QA dataset SQuAD, where given a passage and a question, a system needs to find the exact answer span rather than a sentence. Although SQuAD became very popular and served as a good test set to develop advanced end-to-end neural network architectures, it does not include any nil questions. In practical QA, it is critical to decide whether or not a passage contains a valid answer for a given question. Subsequently, the NewsQA (Trischler et al., 2017) dataset has been released which attempts to overcome this deficiency. However, all the proposed models for NewsQA so far have excluded nil questions during evaluation. Contrary to prior work, we focus on developing models for nil-aware answer span extraction. Very recently, Rajpurkar et al. (2018) released the SQuADRUn dataset by augmenting the SQuAD dataset with unanswerable questions. The unanswerable questions are written adversarially by crowdworkers to look similar to the answerable ones.

7 Conclusion

In this paper, we have focused on nil-aware answer span extraction for RC-based QA. A nil-aware QA system only extracts a span of text from the associated passage as an answer to a given question if and only if the passage contains a valid answer. We have proposed a nil-aware answer span extraction framework based on evidence decomposition and aggregation that can be easily integrated with several recently proposed neural answer span extraction models. We have also developed several pipeline and threshold-based models using advanced neural architectures for comparison. Experiments on the NewsQA dataset show that our proposed framework, when integrated with the answer span extraction models, achieves better performance compared to all the corresponding pipeline and threshold-based models. Employing such a nil-aware answer span extractor in practical IR-style QA tasks will be interesting future work.

References

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model

[7]`https://trec.nist.gov/data/qa.html`

with dynamic-clip attention for answer selection. In *Proceedings of CIKM*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN / Daily Mail reading comprehension task. In *Proceedings of ACL*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of ACL*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in NIPS*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering. In *Proceedings of AAAI*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP*.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017a. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of EMNLP*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017b. ReasoNet: Learning to stop reading in machine comprehension. In *Proceedings of KDD*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, ACL*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using Match-LSTM and answer pointer. In *Proceedings of ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. In *Proceedings of CoNLL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *Proceedings of ICLR*.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *Proceedings of CIKM*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? Fine-grained gating for reading comprehension. In *Proceedings of ICLR*.