On Mining Conditions using Encoder-decoder Networks

Fernando O. Gallego and Rafael Corchuelo
ETSI Informática, Avda. Reina Mercedes s/n., Sevilla E-41012, Spain

Keywords: Condition Mining, Natural Language Processing, Deep Learning, Sequence Labelling.

Abstract: A condition is a constraint that determines when something holds. Mining them is paramount to understanding many sentences properly. There are a few pattern-based approaches that fall short because the patterns must be handcrafted and it is not easy to characterise unusual ways to express conditions; there is one machine-learning approach that requires specific-purpose dictionaries, taxonomies, and heuristics, works on opinion sentences only, and was evaluated on a small dataset with Japanese sentences on hotels. In this paper, we present an encoder-decoder model to mine conditions that does not have any of the previous drawbacks and outperforms the state of the art in terms of effectiveness.

1 INTRODUCTION

A condition describes the circumstances or factors that must be met for something else to hold. Unfortunately, current state-of-the-art text miners do not take them into account, which may easily result in misinterpretations.

For instance, entity-relation extractors (Etzioni et al., 2011; Mitchell et al., 2015) mine entities and relations amongst them and return overviews that are useful to make decisions. For instance, they return ("Acme Bank", "won't merge", "Trust Bank") from a sentence like "May the new law be approved and Acme Bank won't merge Trust Bank"; neglecting the condition might lead a broker not to recommend investing on Acme Bank regardless of the status of the new law. Similarly, opinion miners (Ravi and Ravi, 2015; Schouten and Frasincar, 2016) analyse the opinions that people express in social media, which is very useful for companies to improve their products or to devise marketing campaigns. For instance, they return {flash = -0.99, lens = 0.55} from a sentence like "The flash's horrible indoors, but the lens is not bad for amateurs"; neglecting the conditions might result in a manufacturer not testing the flash appropriately or targeting a campaign to the wrong customers. Analogously, recommenders (Ravi and Ravi, 2015; Schouten and Frasincar, 2016) seek to predict the interest of a user in an item building on his or her previous searches, purchase records, e-mail messages, and the like. Unfortunately, they typically start recommending rent-a-car agents and flights to Australia when they see a sentence like "I'll rent a good car after I book my flight to Australia", independently from whether the user has already booked a flight or not.

The previous examples clearly motivate the need for mining conditions, which has recently attracted the attention of some researchers. The naivest approaches search for user-defined patterns that build on connectives, verb tenses, and other common grammatical landmarks (Mausam et al., 2012; Chikersal et al., 2015). Such a handcrafted approach may result in high precision, but falls short regarding recall because condition connectives have a long-tail distribution, which implies that there are common conditions that do not fit common patterns (see the previous examples). There is only one machine-learning approach (Nakayama and Fujii, 2015), but it must be customised with several specific-purpose dictionaries, taxonomies, and heuristics, mines conditions regarding opinions only, and it was evaluated on a small dataset with 3 155 Japanese sentences on hotels.

In this paper, we present an encoder-decoder approach to mine conditions, which has already proven to be useful in tasks that involve transforming a sequence into another, e.g., machine translation (Cho et al., 2014; Sutskever et al., 2014; Chen et al., 2017), summarisation (Chopra et al., 2016; Nallapati et al., 2016; Rush et al., 2015), or sequence labelling (Ma and Hovy, 2016; Zhai et al., 2017; Zhu and Yu, 2017). Our proposal consists in using two recurrent neural networks (Han et al.,

Gallego, F. and Corchuelo, R.
On Mining Conditions using Encoder-decoder Networks.
DOI: 10.5220/0007379506240630
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 624-630
ISBN: 978-989-758-350-6

2017), namely: the encoder, which encodes the input sentence into a context vector, and the decoder, which decodes the context vector into a sequence of labels that help identify the words that constitute a condition. This approach does not require to provide any user-defined patterns, it does not require any specific-purpose dictionaries, taxonomies, or heuristics, it can mine conditions in both factual and opinion sentences, and it relies on readily-available components (a stemmer and a word embedder). We have performed an experimental analysis on a dataset with 3 790 203 sentences on 15 common topics in English and Spanish; our results prove that our approach beats the others in terms of F_1 score.

The rest of the paper is organised as follows: Section 2 reports on the related work; Section 3 describes our proposal to mine conditions; Section 4 reports on our experimental analysis; finally, Section 5 presents our conclusions.

2 RELATED WORK

(Narayanan et al., 2009) range amongst the first authors who realised the problem with conditions in the field of opinion mining. They devised a feature model that allowed them to machine-learn a regressor that computes the polarity of a conditional sentence, but they did not report on a proposal to mine the conditions. Recently, (Skeppstedt et al., 2015) presented a proposal that helps identify conditional sentences.

The naivest approaches to mining conditions build on searching for user-defined patterns. (Mausam et al., 2012) studied the problem in the field of entity-relation extraction and suggested that conditions might be identified by locating adverbial clauses whose first word is one of the sixteen one-word condition connectives in English; unfortunately, they did not report on the effectiveness of their approach to mining conditions, only on the overall effectiveness of their proposal for entity-relation extraction. Their system was updated recently (Mausam, 2016), but their proposal to mine conditions was not. (Chikersal et al., 2015) proposed a similar, but simpler approach: they searched for sequences of words in between connectives "if", "unless", "until", and "in case" and the first occurrence of "then" or a comma. Unfortunately, the previous proposals are not good enough in practice because handcrafting such patterns is not trivial and the results fall short regarding recall.

The only existing machine-learning approach

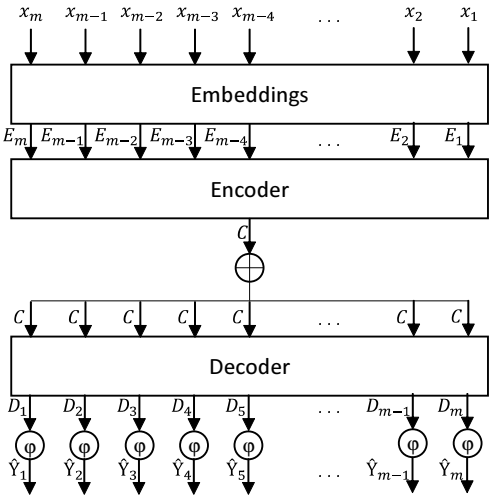


Figure 1: Encoder-decoder model.

was introduced by (Nakayama and Fujii, 2015), who worked in the field of opinion mining. They devised a model that is based on features that are computed by means of a syntactic parser and a semantic analyser. The former identifies so-called bunsetus, which are Japanese syntactic units that consists of one independent word and one or more ancillary words, as well as their inter-dependencies; the latter identifies opinion expressions, which requires to provide some specific-purpose dictionaries, taxonomies, and heuristics. They used Conditional Random Fields and Support Vector Machines to learn classifiers that make bunsetus that can be considered conditions apart from the others. Unfortunately, their approach was only evaluated on a small dataset with 3 155 Japanese sentences regarding hotels and the best F_1 score attained was 0.5830. As a conclusion, this proposal is not generally applicable.

The former approaches confirm that mining conditions is an interesting research problem. Unfortunately, they have many drawbacks that hinder their general applicability. This motivated us to work on a new approach that overcomes their weaknesses and outperforms them by means of a sequence-to-sequence model; our proposal only requires a stemmer and a word embedder, which are readily-available components.

3 MINING CONDITIONS

In this section, we describe our proposal to mine conditions, which consists of an encoder-decoder model that is based on recurrent and bi-directional recurrent neural networks.

Figure 1 sketches its architecture. The input are sentences that are encoded as vectors of the form $(x_m, x_{m-1}, \ldots, x_1)$, where each x_i represents the corresponding lowercased, stemmed word (term) in the sentence ($i \in [1,m], m \geq 1$); note that m must be set a priori to a large enough length and that padding must be used when analysing shorter sentences. The input vectors are first fed into an embedding layer that transforms each term into its corresponding word embedding vector E_i, which is assumed to preserve some similarity amongst the vectors that correspond to semantically similar terms ($E_i \in \mathbb{R}^t$, where t denotes the dimensionality of the word embedding). In order to improve efficiency without a significant impact on effectiveness, we replaced numbers, email addresses, URLs, and words whose frequency is equal or smaller than five by class words "NUMBER", "EMAIL", "URL", and "UNK", respectively. Note that the input vectors encode the reversed input sentences because (Sutskever et al., 2014) suggested that this approach works better with bi-directional recurrent neural networks and does not have a negative impact on regular recurrent neural networks. Furthermore, in order to prevent over-fitting, we used drop-out regularisations (Srivastava et al., 2014) and early stopping (Caruana et al., 2000) when the loss did not improve significatively after 10 epochs. We used the Adam method (Kingma and Ba, 2014) with batch size 32 as the optimiser.

3.1 The Encoder

We decided to implement the encoder using a single-layer neural network, for which we tried two alternatives, namely: a recurrent neural network (RNN) and a bi-directional recurrent neural network (BiRNN). The reason is that these networks are particularly well-suited to dealing with natural language because of their inherent ability to process varying-length inputs (even if they must be encoded using fixed-sized vectors with padding, like in our problem). The difference is that RNNs cannot take future elements of the input into account, whereas BiRNNs can.

Unfortunately, such networks suffer from the so-called exploding and vanishing gradient problems (Bengio et al., 1994; Pascanu et al., 2013). These problems can be addressed by using long-short-term-memory units (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent units (GRU) (Han et al., 2017), which basically help control the data that is passed on to the next training epoch. Our decision was to use GRU units because they seem to be more efficient because they do not have a separate memory cell like LSTM units.

Hereinafter, we refer to the alternative in which we use an RNN with GRU units as GRU and to the alternative in which we use a BiRNN with GRU units as BiGRU. The encoder returns a context vector C that captures global semantic and syntactic features of the input sentences. In the case of the GRU alternative, it is computed as the output of the last GRU unit, that is, $C \in \mathbb{R}^t$; in the case of the BiGRU alternative, it is computed from the last left-to-right GRU unit and the last right-to-left GRU unit, that is, $C \in \mathbb{R}^{2t}$.

3.2 The Decoder

We decided to implement the decoder as a recurrent neural network with four layers since our preliminary experiments proved that this approach was better than using a single layer. We implemented two alternatives, too: one in which we used recurrent neural networks with gated recurrent units (GRU) and another in which we used bi-directional recurrent neural networks with gated recurrent units (BiGRU).

The decoder gets a context vector as input for each recurrent unit of the first layer. Then, it computes an output vector D from each recurrent unit of the last layer. Since the number of recurrent units for each layer is m, the components of the output vector D indicate whether the corresponding input term belongs to a condition or not using the well-known IOB classes, namely: I, which indicates that a term is inside a condition, O, which indicates that it is outside a condition, and B, which indicates that it is the beginning of a condition. The individual components of the output vector are then passed onto a collection of perceptrons that compute the output of our system as follows:

$$\hat{Y}_i = \varphi(W D_i + b) \quad (1)$$

where φ is an activation function, W is a weight matrix, D_i is the output of the decoder, and b is a bias vector. $\hat{Y}_i$ represents the probability distribution of the IOB classes as 3-dimensional vectors. We decided to implement the activation function using either the Softmax function or the Sigmoid function, since the preliminary experiments that we carried out proved that other choices resulted in worse results.

To reconstruct the conditions from this output, we simply take the class with the highest probability and then return all of the subsequences of words in the original sentence that start with a term

with class B that is followed by one, two or more terms with class I.

4 EXPERIMENTAL ANALYSIS

In this section, we first describe our experimental setup, then comment on our results, and finally present a statistical analysis.

4.1 Experimental Setup

We implemented our proposal[1] with Python 3.5.4 and the following components: Snowball 1.2.1 to stemmise words, Gensim 2.3.0 to compute word embedders using a Word2Vec implementation, and Keras 2.0.8 with Theano 1.0.0 for training our neural networks. We run our experiments on a virtual computer that was equipped with one Intel Xeon E5-2690 core at 2.60 GHz, 2 GiB of RAM, and an Nvidia Tesla K10 GPU accelerator with 2 GK-104 GPUs at 745 MHz with 3.5 GiB of RAM each; the operating system on top of which we run our experiments was CentOS Linux 7.3.

We have not found a single record in the literature of a dataset with conditions, which motivated us to create one.[2] It consists of 3 790 203 sentences in English and Spanish that were gathered from the Web between April 2017 and May 2017. The sentences were classified into 15 topics according to their sources, namely: adults, baby care, beauty, books, cameras, computers, films, headsets, hotels, music, ovens, pets, phones, TV sets, and video games. We examined roughly 23 000 sentences and labelled the position where the connectives start, the position where they end, and the position where the conditions end.

We evaluated our proposal on our dataset using 4-fold cross-validation. We set the length of the input vectors to $m = 100$ because the analysis of our dataset revealed that this threshold is enough to deal with the immense majority of sentences that we have found on the Web. We measured the standard performance measures, namely: precision, recall, and the F_1 score. We used the proposals by (Chikersal et al., 2015) and (Mausam et al., 2012) as baselines. The proposal by (Nakayama and Fujii, 2015) was not taken into account because we could not find an implementation, it is not clear whether it can be customised to deal with languages other than Japanese, and the best F_1 attained was 0.5830.

[1]The prototype is available at https://github.com/FernanOrtega/encoder-decoder.

[2]The dataset is available at https://www.kaggle.com/fogallego/reviews-with-conditions.

Table 1: Experimental results.

Lang	Proposal	φ = Softmax			φ = Sigmoid		
		P	R	F_1	P	R	F_1
en	MB	0.6270	0.6144	0.6206	0.6270	0.6144	0.6206
	CB	0.7979	0.4642	0.5870	0.7979	0.4642	0.5870
	Average	*0.7125*	*0.5393*	*0.6038*	*0.7125*	*0.5393*	*0.6038*
	Maximum	*0.7979*	*0.6144*	*0.6206*	*0.7979*	*0.6144*	*0.6206*
	GRU-GRU	0.6816	0.6091	0.6433	0.6455	0.6185	0.6317
	GRU-BiGRU	0.6801	0.6175	0.6473	0.6521	0.6348	0.6433
	BiGRU-GRU	0.6499	0.6181	0.6336	0.6345	0.6091	0.6216
	BiGRU-BiGRU	0.6947	0.5897	0.6379	0.6722	0.6180	0.6440
	Average	*0.6766*	*0.6086*	*0.6405*	*0.6511*	*0.6201*	*0.6351*
	Maximum	*0.6947*	*0.6181*	*0.6379*	*0.6722*	*0.6180*	*0.6440*
es	MB	0.6699	0.5285	0.5909	0.6699	0.5285	0.5909
	CB	0.7953	0.4399	0.5665	0.7953	0.4399	0.5665
	Average	*0.7326*	*0.4842*	*0.5787*	*0.7326*	*0.4842*	*0.5787*
	Maximum	*0.7953*	*0.5285*	*0.5909*	*0.7953*	*0.5285*	*0.5909*
	GRU-GRU	0.6288	0.5683	0.5970	0.6117	0.5641	0.5869
	GRU-BiGRU	0.6392	0.5778	0.6069	0.6664	0.5729	0.6161
	BiGRU-GRU	0.6455	0.5621	0.6009	0.5957	0.5567	0.5755
	BiGRU-BiGRU	0.6411	0.5836	0.6110	0.6349	0.5875	0.6103
	Average	*0.6386*	*0.5729*	*0.6040*	*0.6272*	*0.5703*	*0.5972*
	Maximum	*0.6455*	*0.5836*	*0.6110*	*0.6349*	*0.5875*	*0.6103*

Regarding our proposal, we evaluated eight alternatives that result from combining the two alternatives to implement the encoder (GRU or BiGRU), with the two alternatives to implement the decoder (GRU or BiGRU), and the two alternatives to implement the activation functions in the last layer (Softmax and Sigmoid). We used Categorical Cross-Entropy as the loss function, namely:

$$L(Y,\hat{Y}) = -\frac{1}{m}\sum_{i}^{m}\sum_{j}^{t} Y_{i,j} log(\widehat{Y}_{i,j}) \qquad (2)$$

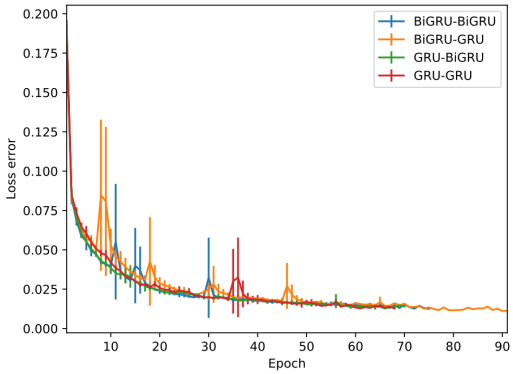
where Y is the expected output vector for a given sentence, $\widehat{Y}$ is the output computed by our system, m is the size of the input vector, and t is the size of the word embedding vectors.

4.2 Experimental Results

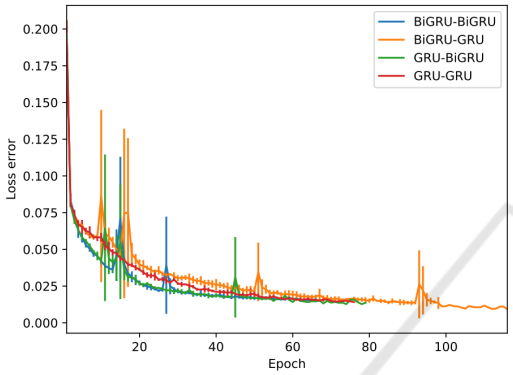
In Figure 2, we show the decay of the loss function during training. Although it decays smoothly for both functions and achieve low loss value, the Softmax activation function seems to decay faster than the Sigmoid activation function.

Table 1 presents the results of our experiments, where MB and CB refer to the baselines by (Mausam et al., 2012) and (Chikersal et al., 2015), respectively. The approaches that beat the best baseline are highlighted in grey.

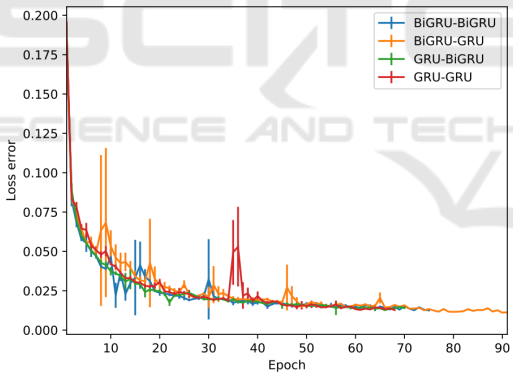
Although the baselines are naive approaches to the problem, they attain relatively good precision; the proposal by (Mausam et al., 2012) attains a recall that is similar to its precision, but the propo-



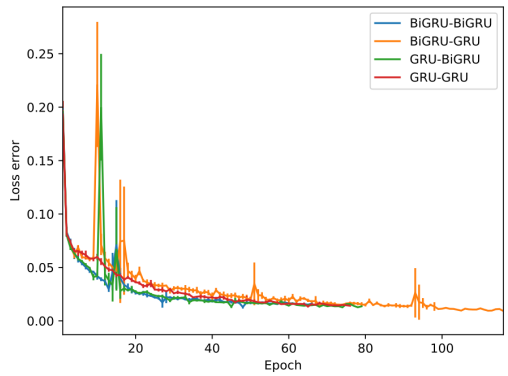
(a.1) Softmax English



(a.2) Sigmoid English



(b.1) Softmax Spanish



(b.2) Sigmoid Spanish

Figure 2: Loss error decay during training epochs.

Table 2: Statistical analysis.

#	Proposal	Comparison	z	p-value
1	GRU-BiGRU	GRU-BiGRU x GRU-BiGRU	-	-
2	BiGRU-BiGRU	GRU-BiGRU x BiGRU-BiGRU	0.6606	0.5089
3	GRU-GRU	GRU-BiGRU x GRU-GRU	1.3212	0.1864
4	BiGRU-GRU	GRU-BiGRU x BiGRU-GRU	1.7340	0.0829

(a) Analysis of our Softmax-based proposals

#	Proposal	Comparison	z	p-value
1	GRU-BiGRU	GRU-BiGRU x GRU-BiGRU	-	-
2	BiGRU-BiGRU	GRU-BiGRU x BiGRU-BiGRU	0.3303	0.7412
3	GRU-GRU	GRU-BiGRU x GRU-GRU	2.4772	0.0132
4	BiGRU-GRU	GRU-BiGRU x BiGRU-GRU	3.3029	0.0010

(b) Comparison of our Sigmoid-based proposals

#	Proposal	Comparison	z	p-value
1	GRU-BiGRU$_{Sigmoid}$	GRU-BiGRU$_{Sigmoid}$ x GRU-GRU$_{Sigmoid}$	-	-
2	GRU-BiGRU$_{Softmax}$	GRU-BiGRU$_{Sigmoid}$ x GRU-BiGRU$_{Softmax}$	0.0826	0.9342
3	MB	GRU-BiGRU$_{Sigmoid}$ x MB	2.2295	0.0258
4	CB	GRU-BiGRU$_{Sigmoid}$ x CB	4.1286	0.0000

(c) Comparison of our winning proposal to the baselines

sal by (Chikersal et al., 2015) falls short regarding recall. None of our approaches beat the baselines regarding precision, but most of them beat the baselines regarding recall since they learn more complex patterns thanks to the deep learning approach that projects the input sentences onto a rich computer-generated feature space. Note that the improvement regarding recall is enough for the F_1 score to improve the baselines.

Regarding the activation function, precision is better for most of our alternatives when the Softmax activation function is used, but all of our alternatives attain similar results regarding recall independently from the activation function. Finally, the best alternatives for English are GRU-BiGRU when using Softmax and BiGRU-BiGRU when using Sigmoid; the best alternatives for Spanish are BiGRU-BiGRU when using Softmax and GRU-BiGRU when using Sigmoid.

4.3 Statistical Analysis

To make a decision regarding which of the alternatives performs the best, we used a stratified strategy that builds on Hommel's test.

In Tables 2.a and 2.b, we report on the results of the statistical analysis regarding our proposal. They show the rank of each approach, and then the comparisons between the best one and the others; for every comparison, we show the value of the z statistic and its corresponding adjusted p-value. In the case of the Softmax activation function, the experimental results do not provide any evidences that the best-ranked approach is different from the others since the adjusted p-value is greater than the

standard significance level $\alpha = 0.05$ in every case. In the case of the Sigmoid activation function, the experimental results do not provide any evidences that the best-ranked approach is different from the second one since the adjusted p-value is greater than the standard significance level; however, there is enough evidence to prove that it is different from the remaining ones since the adjusted p-value is smaller than the significance level. So, we selected the GRU-BiGRU alternative in both cases.

Next, we compared our best alternatives to the baselines. The results of the comparison are shown in Table 2.c, where the subindex denotes the activation function used. According to the statistical test, there is not enough evidence in the experimental data to make a difference between our proposals, but there is enough evidence to prove that they both are significantly better than the baselines.

5 CONCLUSIONS

We have motivated the need for mining conditions and we have presented a novel approach to the problem. It relies on a encoder-decoder model as a means to overcome the drawbacks that we have found in other proposals, namely: it does not rely on user-defined patterns, it does not require any specific-purpose dictionaries, taxonomies, or heuristics, and it can mine conditions in both factual and opinion sentences. Furthermore it only needs two components that are readily available, namely: a stemmer and a word embedder. We have also performed a comprehensive experimental analysis on a large multi-topic dataset with 3 779 000 sentences in English and Spanish that we make publicly available. Our results confirm that our proposal is similar to the state-of-the-art proposals in terms of precision, but it improves recall enough to beat them in terms of the F_1 score. We have backed up the previous conclusions using sound statistical tests.

ACKNOWLEDGEMENTS

The work described in this paper was supported by Opileak.es and the Spanish R&D programme (grants TIN2013-40848-R and TIN2013-40848-R). The computing facilities were provided by the Andalusian Scientific Computing Centre (CICA).

REFERENCES

Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166.

Caruana, R., Lawrence, S., and Giles, C. L. (2000). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NIPS*, pages 402–408.

Chen, H., Huang, S., Chiang, D., and Chen, J. (2017). Improved neural machine translation with a syntax-aware encoder and decoder. In *ACL*, pages 1936–1945.

Chikersal, P., Poria, S., Cambria, E., Gelbukh, A. F., and Siong, C. E. (2015). Modelling public sentiment in Twitter. In *CICLing (2)*, pages 49–65.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In *EMNLP*, pages 103–111.

Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*, pages 93–98.

Etzioni, O., Fader, A., Christensen, J., Soderland, S., and Mausam (2011). Open information extraction: the second generation. In *IJCAI*, pages 3–10.

Han, H., Zhang, S., and Qiao, J. (2017). An adaptive growing and pruning algorithm for designing recurrent neural network. *Neurocomputing*, 242:51–62.

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

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ma, X. and Hovy, E. H. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*, pages 1–11.

Mausam (2016). Open information extraction systems and downstream applications. In *IJCAI*, pages 4074–4077.

Mausam, Schmitz, M., Soderland, S., Bart, R., and Etzioni, O. (2012). Open language learning for information extraction. In *EMNLP-CoNLL*, pages 523–534.

Mitchell, T. M., Cohen, W. W., Hruschka, E. R., Talukdar, P. P., Betteridge, J., Carlson, A., Mishra, B. D., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E. A., Ritter, A., Samadi, M., Settles, B., Wang, R. C., Wijaya, D. T., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2015). Never-ending learning. In *AAAI*, pages 2302–2310.

Nakayama, Y. and Fujii, A. (2015). Extracting condition-opinion relations toward fine-grained opinion mining. In *EMNLP*, pages 622–631.

Nallapati, R., Zhou, B., dos Santos, C. N., Gülçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*, pages 280–290.

Narayanan, R., Liu, B., and Choudhary, A. N. (2009). Sentiment analysis of conditional sentences. In *EMNLP*, pages 180–189.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Ravi, K. and Ravi, V. (2015). A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowl.-Based Syst.*, 89:14–46.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389.

Schouten, K. and Frasincar, F. (2016). Survey on aspect-level sentiment analysis. *IEEE Trans. Knowl. Data Eng.*, 28(3):813–830.

Skeppstedt, M., Schamp-Bjerede, T., Sahlgren, M., Paradis, C., and Kerren, A. (2015). Detecting speculations, contrasts and conditionals in consumer reviews. In *WASSA@EMNLP*, pages 162–168.

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

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Zhai, F., Potdar, S., Xiang, B., and Zhou, B. (2017). Neural models for sequence chunking. In *AAAI*, pages 3365–3371.

Zhu, S. and Yu, K. (2017). Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *ICASSP*, pages 5675–5679.