Multi-Target Stance Detection via a Dynamic Memory-Augmented Network

Penghui Wei, Junjie Lin, and Wenji Mao
[†]SKLMCCS, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[‡]University of Chinese Academy of Sciences, Beijing, China
{weipenghui2016,linjunjie2013,wenji.mao}@ia.ac.cn

ABSTRACT

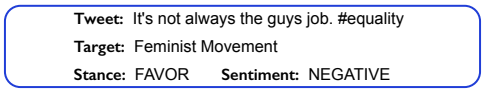
Stance detection aims at inferring from text whether the author is in favor of, against, or neutral towards a target entity. Most of the existing studies consider different target entities separately. However, in many scenarios, stance targets are closely *related*, such as several candidates in a general election and different brands of the same product. *Multi-target stance detection*, in contrast, aims at jointly detecting stances towards multiple related targets. As stance expression regarding a target can provide additional information to help identify the stances towards other related targets, modeling expressions regarding multiple targets jointly is beneficial for improving the overall performance compared to single-target scheme. In this paper, we propose a dynamic memory-augmented network *DMAN* for multi-target stance detection. *DMAN* utilizes a shared external memory, which is dynamically updated through the learning process, to capture and store stance-indicative information for multiple related targets. It then jointly predicts stances towards these targets in a multitask manner. Experimental results on a benchmark dataset show the effectiveness of *DMAN*.

ACM Reference Format:
Penghui Wei, Junjie Lin, and Wenji Mao. 2018. Multi-Target Stance Detection via a Dynamic Memory-Augmented Network. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210145

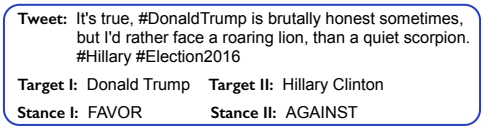
1 INTRODUCTION

People often express their attitudes towards a specific target (e.g., a government policy, a person or a product) through various language expressions. Stance detection aims at determining from text whether the author is in favor of, against, or neutral towards a given target [4]. Automatically identifying user stances by analyzing the online contents they posted has widespread applications in real-life scenarios. For instance, it is beneficial for governments to understand public reactions about policy changes [3]. With the booming of social media, stance analysis in social networking platforms like Twitter has become an important research topic in opinion mining.

Stance detection is a related but different task to aspect-based sentiment analysis (ABSA) which focuses on identifying the sentiment with respect to an entity/aspect. As the example in Figure 1a, the tweet's stance towards the target is supportive but it has negative sentiment, which shows the difference between stance and sentiment. We can also see that the target of interest may not be directly mentioned in a tweet, while in ABSA, aspects are always assumed to be discussed explicitly. Therefore, modeling the relationship between target and tweet is vital for stance detection.



(a) Single-target stance detection.



(b) Multi-target stance detection.

Figure 1: Examples of stance detection tasks.

Most of the existing studies on stance detection treat different target entities separately (i.e., *single-target stance detection*) and ignore the underlying relationship among targets. However, in many scenarios, some target entities are closely *related*, such as several candidates in a general election and different brands of the same product [6], and users often talk about them in the same text. In Figure 1b, the tweet author compares `Donald Trump` with `Hillary Clinton` rather than talks about them separately. The stance expression regarding `Donald Trump` is helpful to identify the stance towards `Hillary Clinton` and vice versa. Hence, utlizing the interrelations among stance expressions is beneficial for improving the overall detection performance compared to single-target scheme.

Jointly detecting the stances expressed in a text towards multiple related targets is defined as *Multi-Target Stance Detection* (MTSD) by Sobhani *et al.* [6]. They propose an attentive encoder-decoder network to capture the dependencies among stance labels regarding multiple targets. The encoder transforms input text into vector representation, and the decoder generates stance words (i.e., FAVOR, AGAINST and NEITHER) for multiple targets one-by-one. Each stance word is generated conditionally on its previous stance words regarding other targets. A limitation of this work is that the order of targets need to be specified in advance (for decoder), and thus the dependencies among stances are "unidirectional", not mutual.

A challenge of MTSD is that we need to extract stance-indicative information concerning multiple targets from a text. However, such information is often implicit and hard to be represented. In recent years, many studies work on augmenting neural networks with *memory* to improve network capacity [1, 8]. They equip neural networks with an auxiliary memory to store useful information that can be accessed via well-designed operations when required.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210145

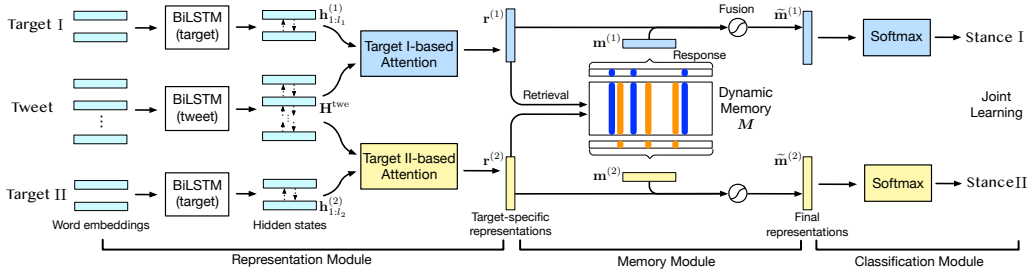


Figure 2: Overall architecture of our dynamic memory-augmented network for multi-target stance detection.

Motivated by this, in this paper, we propose a Dynamic Memory-Augmented Network (*DMAN*) to tackle the problem of MTSD. *DMAN* first learns target-specific text representations for different targets using target content-based attentive network. Afterwards, it utilizes a shared external memory to capture and store stance-indicative information for multiple targets, which is used to enhance the discrimination of text representation. Finally, *DMAN* jointly classifies stances towards these targets in a multitask manner.

The basic idea of equipping the shared memory is to utilize a *unified* module to maintain useful information for *multiple* targets. During the training process, the memory is dynamically updated, aiming at capturing stance-indicative information concerning these targets. By doing this, when predicting stance towards a specific target, *DMAN* accesses the information stored in the memory and adaptively extracts the relevant part for effective detection. Although end-to-end memory networks [8] have been widely applied in ABSA task, the memory used in ABSA models is usually composed of text representation, like word embedding sequence [3] or hidden state sequence of recurrent neural network [9]. In contrast, the memory in *DMAN* is used to maintain implicit information for enhancing text representation.

The contributions of our work are three-fold:

- We propose a novel multitask network *DMAN* to tackle the problem of MTSD in a multitask manner;
- *DMAN* employs a shared external memory to capture and store stance-indicative information for related targets;
- Experimental results on a benchmark Twitter dataset show that *DMAN* achieves the state-of-the-art performance.

2 PROPOSED METHOD

2.1 Problem Formulation

We formularize the problem of multi-target stance detection as follows. Take tweet as an example of text. Given a tweet and a set of K pre-chosen targets, predict the stances towards these targets. We denote the tweet as a word sequence $(w_1, w_2, \ldots, w_L)$, where L denotes the length of it. For the k-th target of K targets ($k \in [1, K]$), we use a word sequence $(w_1^{(k)}, w_2^{(k)}, \ldots, w_{l_k}^{(k)})$ to denote it. Stance labels include FAVOR, AGAINST and NEITHER.

2.2 An Overview of Our Model

Figure 2 illustrates the overall architecture of *DMAN* in the case of $K = 2$. It consists of three modules: (1) Representation module, learning target-specific tweet representations using attentive network; (2) Memory module, equipping a shared external memory to capture and store stance-indicative information for enhancing tweet representations; (3) Classification module, classifying stance towards each target. Next we introduce these modules in detail.

2.3 Learning Target-Specific Representations

In the representation module, we use two bidirectional long short-term memory networks (BiLSTMs) to encode tweet and targets respectively, and then apply target-based attention mechanism to acquire target-specific tweet representation for each target.

2.3.1 ***Tweet and targets encoding.*** We first apply GloVe embeddings pre-trained on Twitter corpus to transform each word w into a real-value vector $\boldsymbol{w} \in \mathbb{R}^{200}$. To effectively utilize contextual features, we apply two BiLSTMs to encode tweet and targets respectively. The BiLSTM for tweet runs over the input tweet $(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_L)$ to produce the hidden states $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_L)$, and we denote this target-independent representation as $\mathbf{H}^{\text{twe}} \in \mathbb{R}^{d \times L}$, where d is output dimension. For each target $k \in [1, K]$, another BiLSTM encodes it and produces $(\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}, \ldots, \mathbf{h}_{l_k}^{(k)}) \in \mathbb{R}^{d \times l_k}$.

2.3.2 ***Target content-based attention mechanism.*** Target content can guide stance detection model to attend the words which make obvious contributions to tweet stance towards this target. Inspired by this and the word-by-word attention model used in question answering [2], we obtain the target-specific tweet representation for each target using an attention mechanism. This mechanism "rereads" the tweet as reading in each target word.

We use the k-th target as an example to describe this mechanism. It first computes the average of the tweet hidden states to obtain the initial representation $\mathbf{o}_0^{(k)} = \frac{1}{L}\sum_{j=1}^{L} \mathbf{h}_j$, and then updates the representation as reading in target words one-by-one. Specifically, when reading in the s-th word of the target (i.e., $\mathbf{h}_s^{(k)}$), the attention module merges it with the previous representation $\mathbf{o}_{s-1}^{(k)}$ and target-independent representation $\mathbf{H}^{\text{twe}}$ to compute importance score of each word in the tweet ($\boldsymbol{e}_s^{(k)} \in \mathbb{R}^{1 \times L}$ in Eq. 1), and normalizes the scores to obtain the attention weight vector $\boldsymbol{\alpha}_s^{(k)}$ (Eq. 2). It then outputs the representation $\mathbf{o}_s^{(k)}$ by combining the previous one $\mathbf{o}_{s-1}^{(k)}$ and the current weight sum of the tweet hidden states (Eq. 3):

$$\boldsymbol{e}_s^{(k)} = \tanh(\boldsymbol{W}_p^{(k)}[\mathbf{h}_s^{(k)}; \mathbf{o}_{s-1}^{(k)}] \otimes \mathbf{1}_L + \boldsymbol{W}_h^{(k)}\mathbf{H}^{\text{twe}}), \quad (1)$$

$$\boldsymbol{\alpha}_s^{(k)} = \text{Softmax}(\boldsymbol{e}_s^{(k)}), \quad (2)$$

$$\mathbf{o}_s^{(k)} = \text{ReLU}(\boldsymbol{W}_o \mathbf{o}_{s-1}^{(k)}) + \mathbf{H}^{\text{twe}} \boldsymbol{\alpha}_s^{(k)\top}, \quad (3)$$

where $\mathbf{1}_L \in \mathbb{R}^L$ is a vector of all 1s, and $\otimes$ is outer product operator. $\boldsymbol{W}_p^{(k)}$, $\boldsymbol{W}_h^{(k)}$ and $\boldsymbol{W}_o$ are trainable weight parameters.

This mechanism models the multiple interactions between tweet and a specific target, accumulating target-specific information by each interaction. We concatenate the latest representation (i.e., $s = l_k$ in Eq. 3) and the average of the target hidden states $\mathbf{h}^{(k)} = \frac{1}{l_k}\sum_{j=1}^{l_k} \mathbf{h}_j^{(k)}$ to obtain the target-specific tweet representation $\mathbf{r}^{(k)}$. Hence, we finally obtain K representations $\{\mathbf{r}^{(k)}\}_{k=1}^{K}$ for K targets.

2.4 Dynamic Memory for Capturing Stance-Indicative Information

Memory-augmented neural networks [1, 8] introduce an auxiliary external memory into networks for improving network capacity and storing task-related information that can be accessed when required. Motivated by this characteristic, we equip our network with an external memory to maintain implicit stance-indicative information that is hard to be represented by the neural network.

2.4.1 ***Dynamic memory updating.*** The shared memory in *DMAN* acts as a knowledge storage that can be used to enhance the discrimination of tweet representation via adaptive operations. As training the model, the memory is dynamically updated together with other parameters in *DMAN*, and becomes more useful. During training, tweet representations regarding multiple targets pass the unified memory, and enable it to capture and hold implicit stance-indicative information concerning these targets. In this way, *DMAN* can effectively leverage the information stored in the memory to enhance tweet representations during inference.

2.4.2 ***Tweet representation enhancement.*** Formally, the external memory is composed of a fix number of unordered memory slices. Each memory slice is a vector, and we denote the external memory as a matrix $\boldsymbol{M} \in \mathbb{R}^{2d \times m}$, where m is the number of slices. The i-th column in $\boldsymbol{M}$ (i.e., the i-th memory slice) is denoted as $\boldsymbol{M}(i) \in \mathbb{R}^{2d}$. The memory matrix is randomly initialized, without using any contents from tweets or targets.

Suppose a tweet representation $\mathbf{r}^{(k)}$ has been obtained, and let it pass the memory. The memory module uses three operations: *retrieval*, *response* and *fusion*, to enhance $\mathbf{r}^{(k)}$ adaptively.

Retrieval of relevant information from memory. The memory module first looks for the pieces of memory slices that are relevant to $\mathbf{r}^{(k)}$ by similarity measure. We use cosine similarity to measure the association degree $w_i^{(k)}$ between $\mathbf{r}^{(k)}$ and a slice $\boldsymbol{M}(i)$:

$$w_i^{(k)} = \frac{\mathbf{r}^{(k)} \cdot \boldsymbol{M}(i)}{\|\mathbf{r}^{(k)}\| \, \|\boldsymbol{M}(i)\|}, \quad i \in [1, m] \,. \tag{4}$$

Generation of response vector. Afterwards, the memory module produces a response vector $\mathbf{m}^{(k)} \in \mathbb{R}^{2d}$ to summarize the relevant information using the weighted sum of memory slices. The weight of a slice $\boldsymbol{M}(i)$ is the normalized association degree $\hat{w}_i^{(k)}$:

$$\hat{w}_i^{(k)} = \frac{\exp(w_i^{(k)})}{\sum_{j=1}^{m} \exp(w_j^{(k)})} \,, \tag{5}$$

$$\mathbf{m}^{(k)} = \sum_{i=1}^{m} \hat{w}_i^{(k)} \boldsymbol{M}(i) \,. \tag{6}$$

Fusion by highway connection. Both $\mathbf{m}^{(k)}$ and $\mathbf{r}^{(k)}$ hold useful information for detecting the stance towards target k. To extract substantial features from them, we employ a highway connection [7] that controls the information flow using gate mechanism:

$$\widetilde{\mathbf{m}}^{(k)} = \boldsymbol{t}^{(k)} \odot \mathbf{m}^{(k)} + (\mathbf{1} - \boldsymbol{t}^{(k)}) \odot \mathbf{r}^{(k)} \,, \tag{7}$$

where $\widetilde{\mathbf{m}}^{(k)} \in \mathbb{R}^{2d}$ is the final representation that will be fed into classification module, and $\odot$ denotes element-wise multiplication. The transform gate $\boldsymbol{t}^{(k)} = \sigma(\boldsymbol{W}_t^{(k)} \mathbf{r}^{(k)} + \boldsymbol{b}_t^{(k)})$ and the carry gate $\mathbf{1} - \boldsymbol{t}^{(k)}$ enable the model to learn how much of $\widetilde{\mathbf{m}}^{(k)}$ is obtained by transforming $\mathbf{r}^{(k)}$ and carrying it, respectively. Here $\sigma(\cdot)$ is Logistic function: $\sigma(\cdot) = \frac{1}{1+\exp(-\cdot)}$. $\boldsymbol{W}_t^{(k)}$ and $\boldsymbol{b}_t^{(k)}$ are trainable parameters.

2.5 Joint Learning Framework

We feed the final representation $\widetilde{\mathbf{m}}^{(k)}$ specific to target k into a fully-connected layer with Softmax function for stance classification. Each target has its own Softmax classifier.

We employ the multitask learning (MTL) framework for MTSD, in which detecting stances towards K targets are treated as K tasks, aiming at improving the generalization of all the tasks by jointly learning them. Two BiLSTMs (one for tweet, another for targets) are shared among all tasks. Let Θ denote the parameter set of *DMAN*, and the goal of the training process is as follows:

$$\min_{\Theta} \sum_{k=1}^{K} \sum_{i=1}^{N} loss(i, k; \Theta) + \lambda \, \|\Theta\|^2 \,, \tag{8}$$

where N denotes the number of training tweets, and $loss(i, k; \Theta)$ is the cross-entropy loss of the i-th tweet in the k-th task. We add ℓ_2-regularization term with coefficient λ to alleviate overfitting.

3 EXPERIMENTS

3.1 Experimental Setup

3.1.1 ***Dataset.*** We conduct our experiments on a benchmark Twitter dataset [6]. Four candidates of the 2016 US election, i.e., `Donald Trump`, `Hillary Clinton`, `Ted Cruz` and `Bernie Sanders`, are the pre-defined targets. Each tweet is annotated with two stance labels regarding two targets (called a target pair). There are three target pairs in this dataset. Table 1 lists the distribution of instances.

Table 1: Statistics of instances in the dataset.

Target Pair	#total	#train	#dev	#test
`Trump-Clinton`	1722	1240	177	355
`Trump-Cruz`	1317	922	132	263
`Clinton-Sanders`	1366	957	137	272
Total	4455	3119	446	890

3.1.2 ***Comparative methods.*** In the literature, *Seq2seq* is the existing state-of-the-art method for MTSD.

- *Seq2seq* [6]: it uses an attentive encoder-decoder network to capture the dependencies among stance labels regarding multiple targets.

Previous studies [5, 6] have shown that *Seq2seq* outperforms a series of methods including some methods for MTSD and the existing state-of-the-art methods for single-target stance detection [4]. Hence, in our experiments, we directly compare our method with the state-of-the-art MTSD work, i.e., *Seq2seq*.

To investigate the effects of different modules in *DMAN*, we conduct ablation test:

- without fusion operation (−F): letting $\widetilde{\mathbf{m}} = \mathbf{m} + \mathbf{r}$ in Eq. 7.
- without dynamic memory (−DM): removing the memory module in *DMAN*.
- without multitask (−MT): the single-task variant that training two models for two targets independently. Each model has a memory module for enhancing tweet representation.
- without dynamic memory and multitask (−DM, −MT): the single-task variant of the above "−DM".
- without dynamic memory, multitask and attention (−DM, −MT, −ATT): two BiLSTMs with average pooling for encoding tweet and target respectively.

Table 2: Performance comparison of different methods.

Method	Target pair			Overall
	Trump Clinton	Trump Cruz	Clinton Sanders	
Seq2Seq	56.60[1]	53.12[1]	54.72[1]	54.81
DMAN	**60.30**	**53.64**	**56.25**	**56.73**
−F	57.94	53.12	54.38	55.15
−DM	55.99	53.22	54.50	54.57
−MT	59.07	52.02	52.90	54.67
−DM, −MT	55.80	51.59	52.26	53.22
−DM, −MT, −ATT	53.55	51.17	46.83	50.52

[1] The results are retrieved from [5].

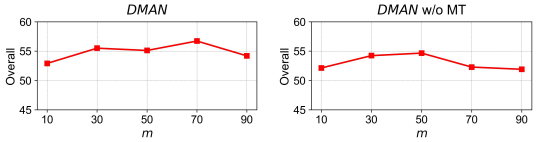


Figure 3: Overall performance w.r.t. memory size m.

3.1.3 ***Evaluation metric (%).*** For each target in a target pair, we compute the average of the F_1 for FAVOR class and the F_1 for AGAINST class, called F_{AVG}. The average of the F_{AVG} for two targets is used as the evaluation metric. The overall performance of three target pairs is computed by the average of them.

3.1.4 ***Implementation details.*** We train all models for max 15 epochs with a batch size of 16 and initial learning rate of 0.001 by Adam. Word embeddings are not updated. d and λ are empirically set to 200 and 10^{-5} respectively. m is set to 70, and the memory matrix $\boldsymbol{M}$ is randomly initialized using standard normal distribution.

3.2 Results and Discussions

3.2.1 ***Comparison of different methods.*** Table 2 summarizes the results of different methods. Comparison between "−DM, −MT, −ATT" and "−DM, −MT" shows that using target-based attention to refine tweet representation is helpful to boost performance. The overall performance of "−DM" outperforms "−DM, −MT" by 1.35%, which indicates that joint training is beneficial for improving generalization, because MTL augments training data implicitly and reduces the number of parameters. The models "−MT" and *DMAN*, which equip a memory module, perform better than "−DM, −MT" and "−DM" respectively. This demonstrates that the external memory can enhance the discrimination of representation. Compared with *Seq2seq*, *DMAN* makes significant improvement. This shows that our method is more effective in jointly modeling stance expressions regarding multiple targets than *Seq2seq*.

3.2.2 ***Influence of memory size*** m. The number of memory slices, i.e., m, controls the capacity of external memory. Figure 3 plots the overall performance of *DMAN* and "−MT" over a set of m values ranging from 10 to 90 with an interval of 20. It can be seen that the performance generally increases at first when m grows, and then it tends to be decreasing. We suggest that too large memory size leads to overfitting and brings more noise information.

3.2.3 ***Effectiveness of external memory.*** To further understand the role of the memory in stance detection, we choose some test tweets of `Trump-Clinton` target pair and Figure 4 shows the visualization of their association degrees $\{\hat{w}_i\}_{i=1}^{70}$ with memory slices. For each target, we choose five tweets with FAVOR label and five tweets with AGAINST label. In each subfigure, a row represents a tweet's association degrees, and a column represents a memory slice. As we can see, for each target, tweets with the same stance label always have higher associate degrees with the same memory slices. This demonstrates that the memory possesses the capacity of capturing stance-indicative information. Further, from the comparison of 4a and 4d, we can find that tweets with FAVOR towards `Trump` and tweets with AGAINST towards `Clinton` have similar associate degree distribution, which accords with our intuition. Similar finding can also be seen in 4b and 4c. These observations indicate that introducing a unified memory module is an effective way to model stance expressions regarding related targets jointly.

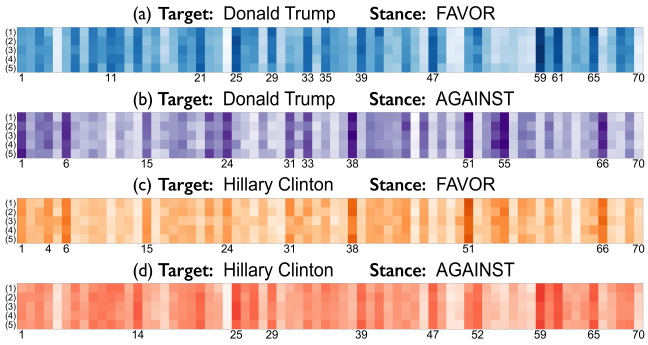


Figure 4: Visualization of association degrees with the memory for 20 test tweets (Deeper color means higher degree).

4 CONCLUSION

In this paper, we propose a novel model *DMAN* for jointly detecting stances towards related targets in a multitask manner. *DMAN* learns target-specific representations by target-based attentive network, and then utilizes an external memory to capture and store stance-indicative information of multiple targets for effective stance detection. Experimental results show the effectiveness of *DMAN*.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grants #2016QY02D0305 and #2016YFC1200702, NSFC Grants #71621002 and #91546112, and CAS Key Grant #ZDRW-XH-2017-3. We thank the anonymous reviewers for the useful comments. We also thank Song Sun for his kind assistance.

REFERENCES

[1] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *arXiv preprint arXiv:1410.5401* (2014).
[2] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of NIPS*. 1693–1701.
[3] Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep Memory Networks for Attitude Identification. In *Proceedings of WSDM*. 671–680.
[4] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting Stance in Tweets. In *Proceedings of SemEval*. 31–41.
[5] Parinaz Sobhani. 2017. *Stance Detection and Analysis in Social Media.* Ph.D. Dissertation. Université d'Ottawa/University of Ottawa.
[6] Parinaz Sobhani, Diana Inkpen, and Xiaodan Zhu. 2017. A Dataset for Multi-Target Stance Detection. In *Proceedings of EACL*. 551–557.
[7] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training Very Deep Networks. In *Proceedings of NIPS*. 2377–2385.
[8] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-End Memory Networks. In *Proceedings of NIPS*. 2440–2448.
[9] Zhao Xu, Romain Vial, and Kristian Kersting. 2017. Graph Enhanced Memory Networks for Sentiment Analysis. In *Proceedings of ECML-PKDD*. 374–389.