Predicting Stance Change Using Modular Architectures

Aldo Porco
Purdue University
aporco@purdue.edu

Dan Goldwasser
Purdue University
dgoldwas@purdue.edu

Abstract

The ability to change a person's mind on a given issue depends both on the arguments they are presented with and on their underlying perspectives and biases on that issue. Predicting stance changes requires characterizing both aspects and the interaction between them, especially in realistic settings in which stance changes are very rare.

In this paper, we suggest a modular learning approach, which decomposes the task into multiple modules, focusing on different aspects of the interaction between users, their beliefs, and the arguments they are exposed to. Our experiments show that our modular approach archives significantly better results compared to the end-to-end approach using BERT over the same inputs.

1 Introduction

One of the main drivers of social interaction is convincing people to adopt new ideas and perspectives. Understanding how to make compelling arguments that would achieve this goal has been studied extensively in psychology and the social sciences (Fogg and B.J., 2003; Popkin, 1991), and more recently by the NLP community. Most of these works focus on analyzing argumentative text in isolation (Habernal and Gurevych, 2016a; Potash and Rumshisky, 2017; Persing and Ng, 2017; Gleize et al., 2019). While the quality of arguments can potentially be judged in isolation based on their merits, *their effectiveness*, when it comes to convincing readers, has to be considered in conjunction with the subjective perspectives and biases of these users. For example, social psychology studies have shown that ideological stances are highly correlated with different moral arguments preferences (Haidt and Graham, 2007; Graham et al., 2009; Graham et al., 2012; Johnson and Goldwasser, 2018), while Lukin et al., (2017) studied the relation between personality type and factual vs. emotional arguments. Durmus and Cardie (2018) studied linguistic properties of convincing arguments, conditioned on users' political and religious convictions.

This work aims to study the effectiveness of debate arguments, with respect to the biases of users reading them. Our goal is to mimic a realistic setting, in which only partial information about these users and their behaviors is available. To accommodate this setting, we use data from the website `debate.org`, used in several previous studies (Durmus and Cardie, 2018; Durmus and Cardie, 2019; Luu et al., 2019; Pacheco and Goldwasser, 2020), and formulate a new classification task, `ChangeMyStance`, (`CMS`)[1], predicting whether a specific user would change their stance after being exposed to two conflicting texts: one that does not support the voting user's views and another one that does. As people do not easily change their minds when exposed to other perspectives, this formulation results in a highly challenging task with only about 6% of users persuaded by the arguments. This formulation is different than the one proposed by Durmus and Cardie (2018), which also studied users' bias, as explained in Section 4.

Our main intuition is that successfully approaching the `CMS` task hinges on modeling the interactions between the arguments made by each side and the biases of the user exposed to them. For example, when debating self-driving cars, a computer scientist might focus on technical challenges while an economist would be more attuned to arguments discussing the implications on the job market. Similar to Durmus

[1]Named after the Reddit sub-community `/r/ChangeMyView` studied by Tan et al. (2016)

Proceedings of the 28th International Conference on Computational Linguistics, pages 396–406
Barcelona, Spain (Online), December 8-13, 2020

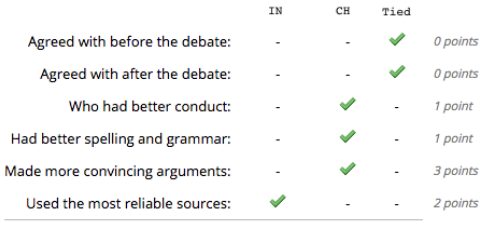

Figure 1: Example debate

Task	Class 0	Class 1	Class 2
AGREEMENT	24310	64122	-
ARGUMENTS	21259	7981	37442
BEFORE	13947	33655	19080
AFTER	13866	35989	16827
CHANGEMYSTANCE	93633	5851	-

Table 1: Data statistics (votes) for each task.

and Cardie (2018), we rely on users' self-reported information, which is often incomplete as users do not report their views in a consistent way and omit details. Instead, *we follow the observation that users' bias can also be reflected by their behavior*, and take a representation-learning approach, mapping the observed users' information to a latent space based on their agreement patterns.

We exploit the fact that the CMS task naturally decomposes into sub-tasks, each capturing a different aspect of the problem. Instead of an end-to-end formulation, we use *modular learning*, in which the different aspects are captured by different modules, combined later to support the final CMS decision. We take a hierarchical task decomposition approach, starting with elementary modules that shape users' representations based on their agreement patterns (AGREEMENT task). We characterize the interaction between users and textual content by defining three auxiliary tasks: characterizing the users' initial bias on the issue, their final stance, and argument preferences (BEFORE, AFTER, and ARGUMENTS, *resp.*).

We consider several ways of using these modules to support the CMS task. The first is the traditional multi-task learning approach (Collobert et al., 2011) in which the input representation is shaped by both the auxiliary and final tasks. *However, the dependency between the* CMS *task and its modules can also build on the output of the modules*. For example, CMS can be approximated by combining the modules for predicting the user's initial bias on the topic and their argument preference (i.e., BEFORE + ARGUMENTS). If the user finds the arguments of the side that conflicts with their original perspective more appealing, they are more likely to be persuaded. We design several hierarchical modular networks, and compare them to a strong end-to-end baseline based on BERT (Devlin et al., 2018). Our results demonstrate the importance of characterizing the user bias using the modules leading to significant improvements.

2 Related Work

Understanding the properties of persuasive arguments and identifying them in text have been widely studied in the NLP community (Tan et al., 2016; Habernal and Gurevych, 2016b; Wei et al., 2016; Persing and Ng, 2017; Stab and Gurevych, 2017; Hidey et al., 2017; Lukin et al., 2017; El Baff et al., 2018; Durmus and Cardie, 2018; Gleize et al., 2019; Yang et al., 2019; Xiao and Khazaei, 2019).

Most relevant to our work is the ChangeMyView (CMV) task, introduced by Tan et al. (2016), predicting stance changes in Reddit forums. Similar to our work, Durmus and Cardie (2018) highlight the importance of modeling the user's belief when analyzing persuasive language. In their work, Xiao et al. (2019) approach the CMV problem using features that characterize the users' psychological attributes.

We suggest a modular approach, adapting the modular architecture suggested by (Zhang and Goldwasser, 2019) for sequence tagging to detecting user-specific persuasive text. Closest to our settings is Jo et al. (2018), which follows the CMV line of research and proposed an end to end neural network with a modular attention mechanism. It consists on two modules, one that models the malleable parts of the original poster's arguments, and another that focuses the difference in content between both authors.

3 The Dataset and Preprocessing

The ChangeMyStance task is defined over two texts with opposing perspectives on an issue, and a user indicating if their stance on the issue changed after reading the arguments in the texts. The debates, collected from www.debate.org, are between two contenders: the initiator (IN) proposing the debate question and the challenger (CH), offering the opposing view. Debates consist of a series of rounds, in

which contenders respond to the arguments made by the other contender in the previous round. At the end of the debate, users can express their preferences by voting, (as depicted in Fig. 1) by casting a 0 (vote for IN), 2 (voter for CH) or 1 (tied) in the following categories: $\mathtt{U}_{\mathtt{before}}$ (voter's preference **before** the debate), $\mathtt{U}_{\mathtt{after}}$ (voter's preference **after** the debate.), $\mathtt{U}_{\mathtt{arguments}}$ (Which side made better **arguments**?), $\mathtt{U}_{\mathtt{conduct}}$ (Which side had better **conduct**?), $\mathtt{U}_{\mathtt{writing}}$ (Which side had better **writing** skills?).

Users can share information on their profile page, including attributes such as age, political ideology and education. They can also provide their stances over 48 "big"-issues (e.g., abortion, healthcare, etc.), and provide an open-ended textual summary. We refer to the first two (*big issues* stances and profile attributes) as $\mathtt{U}_{\mathtt{profile}}$ and use $\mathtt{U}_{\mathtt{summary}}$ to refer to the textual summaries, represented by applying a pre-trained BERT encoder on that text.

Preprocessing We filter debates containing arguments by only one side, or low engagement (less than 3 votes). After filtering we are left with 12901 debates and 4380 different voters participating in them.

We represent the debate text by fine-tuning the BERT (Devlin et al., 2018) model on our data, treating each round as a sentence capped to a size of 510 tokens. In order to represent the text of the posts we use a fixed BERT encoding of the first 510 tokens of the debate (same for $\mathtt{U}_{\mathtt{summary}}$).

4 Problem Definition and Decomposition

In this section we will define the ChangeMyStance task, its supporting sub-problems, and explain how it differs from similar persuasion-detection tasks.

ChangeMyStance is a binary classification problem, each instance is a 4-tuple $< \mathtt{U}, \mathtt{T}_1, \mathtt{T}_2, \mathtt{y} >$, where U is a user representation, $\mathtt{T}_1$ and $\mathtt{T}_2$ are two pieces of text with opposing views. As a convention, we assume that $\mathtt{T}_2$ is the text expressing U's initial stance on the topic. The label $\mathtt{y} \in \{0, 1\}$ captures the stance change, i.e., $\mathtt{y} = 1$ if $\mathtt{T}_1$ made the user change their mind, i.e., when $\mathtt{U}_{\mathtt{before}} \neq \mathtt{U}_{\mathtt{after}}$. When the user is initially undecided, i.e., $\mathtt{U}_{\mathtt{before}} = 1$, we allow both sides to change the user stance. Changes to an undecided view are also considered as stance changes.

Comparison with Durmus and Cardie (2018) The interaction between users' bias and the type of arguments they find convincing over the debate.org dataset was previously studied (Durmus and Cardie, 2018). They defined controlled studies in which the users were known to change their mind, or required models for predicting the stance preference rather than stance change. *However, our task formulation is very different*. We focus on modeling the realistic setting of persuasion detection, in which only a small fraction of users will be convinced to change their mind. In our view, this is a more natural setting. For example, when launching a campaign on social media promoting better health choices, policy makers have to consider which arguments will be convincing based on limited information. We specifically focused on these challenging settings, as a way to evaluate different strategies to represent user's beliefs and its interaction with argumentative content.

4.1 Decomposing ChangeMyStance

Our main technical insight is that the CMS task can be decomposed into several sub-tasks, which characterize the users' biases and preferences. These subtasks are conceptually simpler to learn, and help structure the learning process for the downstream CMS by providing intermediate representations for it. We begin by describing these tasks and their intended contribution, and then describe the modular architecture we used in Sec. 5. The data statistics of each task can be found in Tab. 1.

Sub-task: ARGUMENTS This task is designed to characterize which argument the user finds more appealing. Note that it is different than CMS, as users are primed to pick arguments that support their existing views. The problem is defined over the same representation as CMS, $\langle \mathtt{U}, \mathtt{T}_1, \mathtt{T}_2, \mathtt{y} \rangle$, with $\mathtt{y} = \mathtt{U}_{\mathtt{arguments}}$.

Sub-task: BEFORE This task is designed to characterize users' initial biases, by predicting which argument ($\mathtt{T}_1$ or $\mathtt{T}_2$) aligns with the voter initial beliefs, i.e., $\mathtt{U}_{\mathtt{before}}$. It takes similar inputs as CMS. While CMS instances already capture this information in their argument order ($\mathtt{T}_2$ captures their initial bias), this task allows the model to tune the user representation to capture this initial bias.

Sub-task: AFTER This is a multi-class classification problem, in which the label corresponds to the voter beliefs after attending the debate. The problem has same sample representation as CMS $\langle \mathtt{U}, \mathtt{T}_1, \mathtt{T}_2, \mathtt{y} \rangle$, but $\mathtt{y} = \mathtt{U}_{\mathtt{after}}$, thus $\mathtt{y} \in 0, 1, 2$.

Sub-task: AGREEMENT This task is designed to shape the users' representation, by increasing the similarity between users with similar beliefs (i.e., before votes). This task is particularly useful as it helps the model overcome the problem of sparse user information. An agreement instance, $\langle \mathtt{U}_1, \mathtt{U}_2, \mathtt{y} \rangle$, has a positive value ($\mathtt{y} = 1$) if the users have the same BEFORE vote, and a negative value otherwise.

5 Models

In the previous section we introduced the CMS task, and four related sub-tasks, designed to support it. We use the term **module** to refer to the neural architecture trained for each one of these tasks, and view it in a dual way - as a *classifier*, by providing a probability distribution over the output labels for a given input, and as a *representation extractor*, providing an embedding of the inputs, capturing the specific aspect of the sub-task (e.g., embedding the user's initial bias on a topic), using the module's final hidden layer.

Our main technical challenge is to design a learning and reasoning framework that can combine the supporting tasks and the final CMS prediction. We explore three approaches for this purpose:

1. *Multitask Learning*: similar to Collobert et al. (2011), we jointly train all the modules, while sharing the input representation parameters.
2. *Ensemble-based*: constrain the modules' *outputs* to be consistent with the CMS prediction, by defining an ensemble of multiple modules predicting the same output.
3. *Modular-Architecture*: most of our efforts focus on exploring modular architectures, which combine pre-trained modules into a hierarchical neural architecture. In this way, the modules provide a supporting structure, using their final layer activation as a representation for the parent module.

Intuitively, both the ensemble-based and the modular approaches exploit the sub-tasks modules. The first looks at the modules' outputs, while the second uses the modules as representation extractors, incorporated into downstream tasks representations, and adapted during the training process.

In the rest of this section we explain the different choices. We begin with the basic modules for each task, trained in an end-to-end fashion (Sec. 5.1). Then, in Sec. 5.2, we explain our hierarchical modular approach. Finally, we explain the multi-task (Sec. 5.3) and ensemble-based (Sec. 5.4) approaches.

5.1 Atomic Modules

Our model connects multiple modules hierarchically, building on atomic (leaf) modules, defined over the raw inputs, to create higher level modules. We begin by describing the architecture of atomic modules.

User Representation and AGREEMENT Module Characterizing the users and their biases is at the heart of our work. Past work looked at personality types (Lukin et al., 2017), and broad stances on issues (Durmus and Cardie, 2018) as a way to do that. In this work we take a representation learning approach, and learn a user representation based on both the user's profile information and their behavior, used to shape a latent user representation using the AGREEMENT module. The basic user representation consists of three elements: $\mathtt{U}_{\mathtt{profile}}$ (profile attributes and *big issues* stances), $\mathtt{U}_{\mathtt{summary}}$ (BERT encoding of the user summary) and $\mathtt{U}_{\mathtt{emb}}$ (randomly initialized user embedding).

The module associated with the AGREEMENT task is designed to shape this representation, to capture the similarity in perspectives and biases of voters who have the same initial views on topics. Since many users only provide partial information in their profile, this module implicitly captures the relationship between user properties and initial biases, and helps complete missing information.

The architecture of this module, depicted in Fig. 2, first encodes the user summary $\mathtt{U}_{\mathtt{summary}}$ using two feed-forward layers. Then it concatenates it together with the $\mathtt{U}_{\mathtt{emb}}$ and the profile features $\mathtt{U}_{\mathtt{profile}}$. All this information is then passed through a user encoder of two more hidden layers. We use the embedding hinge loss objective to quantify the similarity error. To reduce noise, we ignore "tie" votes.

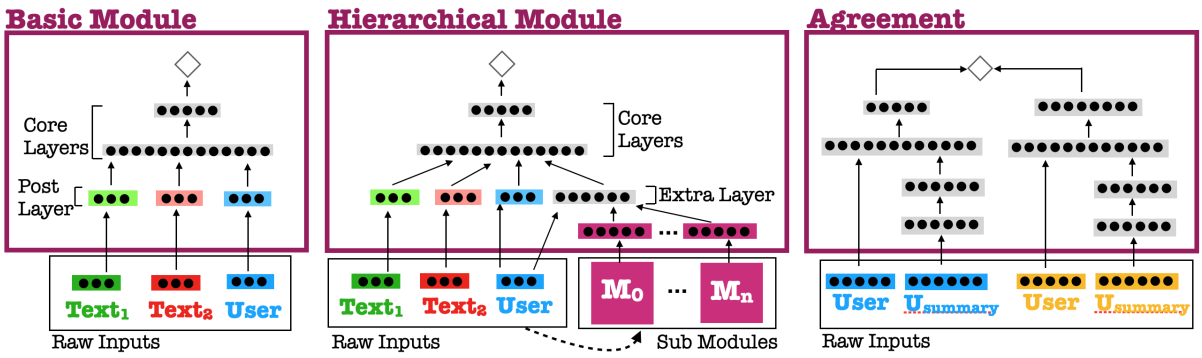

Figure 2: Module types.

Modules for User and Text Interaction The other four tasks (CMS, ARGUMENTS, BEFORE, AFTER) defined in Sec. 4 aim to characterize the interaction between a user and text. As a result, all tasks take the same inputs, and use the same architecture (instantiated with different parameters for each module). We refer to the atomic version of these modules as **basic modules**. Fig. 2 describes the architecture:

- A *post layer* that transforms $\texttt{T}_1$ and $\texttt{T}_2$ BERT encoding separately to a reduced representation.
- Two *core layers* that take as input the concatenation of the $\texttt{T}_1$ hidden representation, $\texttt{T}_2$ hidden representation and the `User` representation.

We use the Cross Entropy as loss function for these multi-class classification problems.

5.2 Hierarchical Models

The atomic modules can be used as building blocks for more complex architectures. We leverage the modules' role as *representation extractors* and use their final core layer as their representation. We pre-train the modules to obtain this representation.

The architecture of the hierarchical modules are defined recursively. It extends the architecture of the basic module defined above, by adding an additional input $\texttt{h}_\texttt{M}$, which represents the embedding of all modules[2]. Given a hierarchical module h, we define the set of supporting modules as $M(h)$, each $m \in M(h)$ corresponds to the output layer of m. We define a hidden representation for each module, which adapts during the training process: $e_m = f(W_m^0\ m + b_m)$, where W_m^0, b_m are the weight matrix and bias term associated with the module instance m, and f is a non-linear activation function. Now we can define $M(h)$, the additional input representing the supporting modules–

$$M(h) = f\left(\sum_{m \in M(h)} W_m^1\ \ e_m + b_{e_m}\right)$$

This architecture is described in Fig. 2. Note that each sub-module $m \in M(h)$ can be hierarchical as well. We will use the following nomenclature to describe the recursive structure: $\texttt{M}[\texttt{SubM}_1, ..., \texttt{SubM}_\texttt{n}]$, where `M` is the name of the topmost hierarchical module, and $\texttt{SubM}_\texttt{i}$ are sub modules. This structure can be recursive. E.g, $\texttt{CMS}[\texttt{CMS}[\texttt{Aft}[\texttt{Ag}], \texttt{Bef}[\texttt{Ag}]], \texttt{CMS}[\texttt{Bef}[\texttt{Ag}], \texttt{Args}]]$ corresponds to a hierarchical `CMS` module that has two `CMS` sub-modules: the left one has $\texttt{After}(\texttt{Aft})$ and $\texttt{Before}(\texttt{Bef})$ sub-modules, while the right one has `Bef` and $\texttt{Arguments}(\texttt{Args})$ sub-modules. The sub-modules are also hierarchical and they use $\texttt{Agreement}(\texttt{Ag})$ as a sub-module.

User Representation In the architecture the user representation is used in two ways, it is concatenated to the output of the post layer, and it is also added as an additional sub-module to $M(h)$. The motivation behind it is that the user representation can balance the importance of each module. For example, when trying to predict `CMS` using `Before` and `Arguments` as supporting modules, an specific user may give more value to a sound argument than to their previous beliefs.

[2]We add the input user representation as another sub-module to all hierarchical modules, omitted from the notation to help clarity. See Fig. 2 and user representation discussion.

5.3 Multitask Learning

We also experimented with a joint model where all tasks are learned at the same time. First, we have a shared debate encoder that: (1) takes the raw inputs and passes them through a feed-forward layer (like the Basic Model), then (2) they are concatenated and passed trough two feed-forward layers. Separately, for each task we have two feed forward layers plus one last $Softmax$ layer that are fed from the shared debate encoder. We use the sum of the Cross Entropy Losses from each task as the learning objective.

5.4 Ensemble based Prediction

The sub-tasks we introduced provide an alternative representation for the CMS task. In the modular-architecture approach, this representation is captured by the neural information flow. In this section we look at a different way to use the modules, by forcing consistency over the *outputs* of these modules. Given an instance x, we can define the task over the outputs of the modules as follows–

(1) ARGUMENTS $\neq$ BEFORE $\implies$ ChangeMyStance
(2) AFTER $\neq$ BEFORE $\implies$ ChangeMyStance
(3) End2End CMS $\implies$ ChangeMyStance

The rules are defined over the outputs of the corresponding modules, and capture the conditions in which different output assignments to the modules indicate a change in stance. Rule 1 states that if the side with better arguments is not the same side as the initial bias, a stance change happens. It is captured by the output of the module CMS[Before[Agree], Args]. Rule 2, looks at the difference between the view before and after the debate, inconsistent values reflect and a stance change. This rule is captured by the output of the module CMS[Before[Agree], After[Before[Agree], Args]]. Finally, rule 3, uses the prediction of the end-to-end CMS module. We combine the prediction of the modules using an ensemble approach, where each "expert" contributes a prediction and a confidence score, normalized into a probability. We sum the scores associated with each label and predict the highest scoring output over all three predictions. We tune the relative weights of these predictions using the validation data.

6 Experimental Design

We run each setting using a 10-fold cross validation. For each fold the data was separated by debates, i.e. all votes of the same debate are associated with the same fold. In each run, one fold was used for testing and the rest for validation and training. We randomly chose 15% (20% for *Arguments*) of the training as validation data keeping votes of the same debates together. We use the Adam optimizer with a 256 batch size and the Sigmoid activation function. Due to the high class imbalance, we balance the weight of the classes in the objective function. For the ARGUMENTS, AFTER, BEFORE and CHANGEMYSTANCE modules we use a learning rate of 0.0001. Training stops if there is no improvement after 25 epochs or 200 epochs have passed. *Agreement* uses a learning rate of 0.001 and stops after 50 epochs of no improvement.

Raw Inputs The representation of each debate stance is the BERT encoding of the first 510 characters (9216 features). This representation decision helped limit the computational burden, however it can potentially lead to information loss, as in some cases the combined arguments of each stance can be longer, which was the case in 50% of the debates. We ran two experiments to validate this design decision, comparing the E2E CMS model trained over the truncated text to two systems trained over the full text. The first was a bag of words of the top 20,000 frequent unigrams, resulting in a 6% in +F1 score drop. The second model used a separate BERT encoding of each debate round, and the full stance representation was created by averaging these vectors. This representation imposed a higher computational cost, however it did not lead to a statistically significant result difference compared to the truncated version.

The user profile $U_{profile}$ has 158 features by putting together demographic information and *big issues*. The randomly generated user embedding U_{emb} is 100D (50D for *Arguments*). The user summary $U_{summary}$ was used only as part of the *Agreement* module and it has the same number of features as the BERT encoding.

Task	Text	+ $U_{profile}$	+ U_{emb}
	Avg. F1	Avg. F1	Avg. F1
ARGUMENTS	0.4913	0.494	0.507*
BEFORE	0.4264	0.464*	0.517*
AFTER	0.4389	0.475*	0.534*
	+F1	+F1	+F1
CMS	0.170	0.197*	0.272*

Table 2: **Evaluating Different User Representations**. Avg. F1-score and +F1 (positive class F1) for the E2E model using different user representations. All numbers are truncated to the 3rd decimal. We test statistical significance "*" with $p - value < 0.01$ over the closest simplified version. For Example: "+ U_{emb}" is tested against "+ $U_{profile}$".

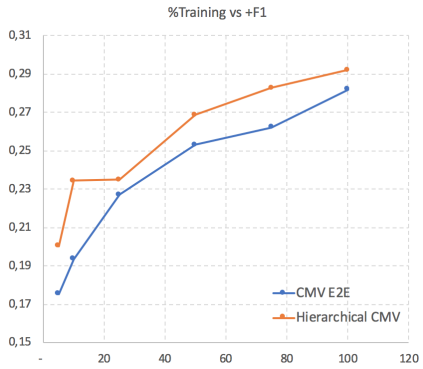

Figure 3: Percentage of training samples used vs. CMS's positive class (+F1) when using the E2E CMS and the best Hierarchical version.

Neural Architectures:
(1) *Hierarchical and Basic:* all modules' architectures include two core layers, a post Layer and an extra layer (shown in Fig. 2). The post layer is 100D, the extra layer is the same dimensionality as its input (varies depending on the number of supporting modules). Core layers consist of two 50D FF layers, i.e., each sub-module contributes 50 features to their parent module.
(2) *Agreement:* uses a different sample structure (two users) and outputs their embedding. When used as a sub-module it returns only the first user embedding. It consists of two 100D core layers and two 50D summary layers, described in Fig. 2.
(3) *Multitask:* we compare the multitask approach to the hierarchical model, that uses the same sub-modules. The multitask architecture resembles the Basic Module (Fig. 2) structure. The post layer is shared by all the tasks, while each task individually manages their own core layers. As before, we sum the tasks' objectives. Samples from all tasks are shuffled and an epoch processes a batch 256.
(3) *Ensemble:* In order to choose the weights between rules we tested all combinations with a 10% weight difference over the validation set and chose the one with the best performance.

7 Results

7.1 The importance of user representation

Our first set of experiments demonstrate the importance of creating rich user representations. We compared several different representations when learning the CMS task, using the basic module. First we started from no user representation (Text), followed by including the $U_{profile}$ and finally adding the randomly initialized user embedding (U_{emb}).

Our results can be found in Table 2. They show that in all tasks increasing the complexity of the user's representation improves performance. Although most profiles only have little information, the $U_{emb} + U_{profile}$ setting shows they are enough to produce a 4 and 3.5 points increase in the BEFORE and AFTER task respectively. Moreover, it manages to increase the F1-score for the positive CMS task by almost 3 points and its average F1-score by 4 points. Also, we can see that the increase is bigger in tasks where the users beliefs are key (BEFORE and AFTER). As expected, the ARGUMENTS task is less sensitive to the user representation, and depends more on the text.

The user representation can be fine-tuned using the AGREEMENT module, that takes the representation of two users that expressed the same prior belief and aligns their representations. Our hypothesis is that this module helps characterize users that have sparse profile information and have only contributed a few votes. It aligns the representations of highly engaged users who vote frequently with the low engagement users, based on their initial bias on the debate issue, resulting with a highly tuned representation. As can be seen in Table 5, the AGREEMENT module produces statistically significant results when used as part

Models	Val.	Test
Baselines	+F1	+F1
Random CMS	-	0.108
E2E CMS (Text + $U_{profile}$)	0.203	0.197
(1) E2E CMS (Text + $U_{profile}$ + U_{emb})	0.282	0.272
Traditional Approaches	+F1	+F1
(2) Multitask[CMS, BEFORE, AFTER, ARGS]	0.283	0.278
(3) Ensemble	0.283	0.279^{*}
Hierarchical Models	+F1	+F1
(4)CMS[BEFORE, AFTER, ARGS]	0.297	$0.288^{\cdot}$
(5) CMS[AFTER, BEFORE, ARGS, AGREE] ,	0.296	0.286^{*}
(6) CMS[CMS[BEFORE[AGREE], AFTER[BEFORE[AGREE], ARGS]], CMS[BEFORE[AGREE], ARGS]]	0.293	0.284^{*}
(7) CMS[AFTER[BEFORE[AGREE], ARGS], BEFORE[AGREE], ARGS, AGREE]	0.299	0.292^{+-}

Table 3: Results of using different strategies when shaping the modules interaction to predict CMS. We use the *+F1* (the F1-score of the positive CMS class) metrics to characterize the models. We test for statistical significance with $p - value < 0.05$: (1) "*" w.r.t E2E CMS (Text + $U_{profile}$ + U_{emb}), (2) "$\cdot$" w.r.t Multitask, (3) "$^{+}$" w.r.t Ensemble, and (4) "$^{-}$" w.r.t CMS[AFTER, BEFORE, ARGS, AGREE].

Ideology		Metric		
Voter	Writer	%+	+F1	Avg. F1
Liberal	Liberal	7.8	0.279	0.600
Liberal	Conserv.	3.7	0.289	0.602
Conserv.	Liberal	6.6	0.293	0.607
Conserv.	Conserv.	15.9	0.405	0.647

Table 4: Predicting change of stance based on ideology agreement of Conservatives and Liberals. The study uses as metrics the percentage of the positive class (%+), the F1-score of the positive class (+F1) and the Avg. F1-score.

Models	Val.	Test
	Avg. F1	Avg. F1
E2E AGREEMENT	0.528	0.520
Random ARGUMENTS	-	0.398
E2E ARGUMENTS	0.534	0.507*
Random BEFORE	-	0.415
E2E BEFORE	0.531	0.517*
M.Bef: BEFORE[AGREE]	0.544	0.533*
Random AFTER	-	0.410
E2E AFTER	0.548	0.534*
M.Aft: AFTER[ARGS, BEFORE[AGREE]]	0.562	0.550*

Table 5: Avg. F1 scores of the end-to-end (E2E) and Hierarchical models for the supporting tasks. If $p - value < 0.01$ then * when comparing basic and hierarchical. We use AGREE and ARGS as short names for AGREEMENT and ARGUMENTS respectively.

of the BEFORE module, and it is used by our top performing model on the CMS task, as shown Table 3.

We provide additional analysis based on users' ideology in Table 4. We compare our model's performance over different data splits based on the ideology values of the debating user and voting user. Interestingly, the task is significantly easier when both users has a conservative ideology.

7.2 Evaluating Hierarchical Modules on CMS

Can the selected sub-tasks help predict CMS? First we want to show that using the BEFORE, AFTER, ARGUMENTS and AGREEMENT modules is relevant for the CMS task. We compare the end-to-end version basic module, with several hierarchical systems. First, we compare it with the simplest hierarchical model, CMS[Bef, Aft, Args, Agree] (model 5). As can be seen in Table 3, our two level hierarchy model (5) performs statistically significantly better than the end to end CMS model (1).

What is the right modular structure? Constructing hierarchical structures allows us to encode domain knowledge into the representation. Based on our experiments we suggest several modules hierarchies.

- BEFORE should use AGREEMENT as sub-module as understanding which users have similar beliefs is critical for predicting biases. As shown in Fig. 5, the modular version of the BEFORE module (M.BEF) is significantly better by 1.5 points.
- The beliefs AFTER the debate are a consequence of how strong the voter's bias is, and the ARGUMENTS used to persuade the user. Fig. 5 shows that M.AFT is significantly better than its end to end version AFTER.
- As the problem is defined, if AFTER is different than BEFORE it means the user changed their stance. Therefore, BEFORE and AFTER should always support CMS. As can be seen in Table 3 all the CMS models that build on these two sub-tasks significantly outperform the end to end model.

Model	Accuracy
Majority Vote TASK1	0.5
Durmus Model TASK1	0.657
E2E TASK1	0.642
E2E TASK1[BEFORE, AFTER, ARGUMENTS]	0.659*

Table 6: Results obtained using the modular approach on Task 1 (debates in all categories) of Durmus el al. (Durmus and Cardie, 2018) paper. This version has 4392 samples split evenly. (*) indicates statistical significance ($p-value < 0.05$) when comparing E2E TASK1 and the corresponding hierarchical version.

Building on domain knowledge as a heuristic to shape the representation, we evaluate several modular hierarchies. One option is to put all modules at the same level and let them share information (model (5) in Table 3). Another option is to use a deeper hierarchy to support the CMS problem (model (7) of Table 3), which works significantly better. In other words, building hierarchies based on improved sub-modules can improve the overall performance.

We also evaluate model-7 when less supervision is available. These settings aims to evaluate whether the modules can act as scaffolding, allowing the model to leverage the limited supervision in a better way. We evaluate these settings by training the AGREEMENT, BEFORE, AFTER, ARGUMENTS modules using all the data, and increase the amount of CMS supervision to form a learning curve. The results summarized in Fig. 3 show the modular approach consistently outperforms the E2E one.

Modular Learning vs. other information sharing approaches As shown in Figure 3, both the multitask model (2) and the ensemble model (3) achieve a limited improvement over the E2E model and perform worse than the simplest hierarchical model (4) that uses the same tasks (model (6), ensemble).

7.3 Predicting Stance Change Direction

The CMS task assumes knowledge about the initial stance of the user (encoded by the order of the two input texts), and predicts whether it changed. In Durmus and Cardie (2018), a different task was studied. They assumed that the stance change is known, but the direction of change is not, i.e., whether the voting user was convinced by the PRO debater or CON debater.

We build their task and evaluated our hierarchical module on it. Unfortunately, we were not able to directly compare our work with theirs since the original debates splits were not available. Table 6 shows the results of reproducing their setting (Task-1, all the categories) using our models and experimental design settings. Our modular approach improves the E2E model by 2%, similar to our CMS problem. We included the results achieved by Durmus et al. (Durmus and Cardie, 2018) for reference, but they are not comparable as their dataset evolved and the splits are not the same.

8 Conclusions and Future Work

In this paper we study the problem of identifying whether debate arguments will convince their reader to change their minds. We focus on a highly challenging version of this task in which only 6% of the users actually change their mind, requiring models that can capture the interaction between users, their biases and relevant arguments for them. These settings, while challenging, are natural - we often would like to predict the impact of different arguments on different demographics, allowing policy makers to adapt their arguments and help people make better choices.

We propose a hierarchical modular learning approach, which relies on decomposing the stance-change task into different aspects, such as characterizing the users biases and the quality of the arguments made. We describe a general formulation for composing these modules recursively. Our approach can be considered as a form of *scaffolded learning*, in which we construct increasingly complex task representations in a hierarchical way, by using the previously learned modules to construct a higher level representation, or scaffolding, supporting the new learning task. Our experiments show the strength of this approach, raising directions for future research on leveraging modular architectures on a larger scale, to capture different forms of users' behaviors and characterize the arguments in more nuanced ways.

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 12, nov.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Esin Durmus and Claire Cardie. 2018. Exploring the role of prior beliefs for argument persuasion. pages 1035–1045.

Esin Durmus and Claire Cardie. 2019. Modeling the factors of user success in online debate. In *The World Wide Web Conference*, pages 2701–2707. ACM.

Roxanne El Baff, Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. 2018. Challenge or empower: Revisiting argumentation quality in a news editorial corpus. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 454–464, Brussels, Belgium, October. Association for Computational Linguistics.

B. J. Fogg and B.J. 2003. *Persuasive technology : using computers to change what we think and do*. Morgan Kaufmann Publishers.

Martin Gleize, Eyal Shnarch, Leshem Choshen, Lena Dankin, Guy Moshkowich, Ranit Aharonov, and Noam Slonim. 2019. Are You Convinced? Choosing the More Convincing Evidence with a Siamese Network.

Jesse Graham, Jonathan Haidt, and Brian A Nosek. 2009. Liberals and conservatives rely on different sets of moral foundations. *Journal of personality and social psychology*, 96(5):1029.

Jesse Graham, Brian A Nosek, and Jonathan Haidt. 2012. The moral stereotypes of liberals and conservatives: Exaggeration of differences across the political spectrum. *PloS one*, 7(12):e50092.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223, Austin, Texas, November. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016b. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

Jonathan Haidt and Jesse Graham. 2007. When morality opposes justice: Conservatives have moral intuitions that liberals may not recognize. *Social Justice Research*, 20(1):98–116.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21.

Yohan Jo, Shivani Poddar, Byungsoo Jeon, Qinlan Shen, Carolyn Rose, and Graham Neubig. 2018. Attentive Interaction Model: Modeling Changes in View in Argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 103–116, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kristen Johnson and Dan Goldwasser. 2018. Classification of moral foundations in microblog political discourse. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 720–730.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument strength is in the eye of the beholder: Audience effects in persuasion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 742–753.

Kelvin Luu, Chenhao Tan, and Noah A Smith. 2019. Measuring online debaters' persuasive skill from text over time. *Transactions of the Association for Computational Linguistics*, 7:537–550.

Maria Leonor Pacheco and Dan Goldwasser. 2020. Modeling content and context with deep relational learning. *arXiv preprint arXiv:2010.10453*.

Isaac Persing and Vincent Ng. 2017. Why can't you convince me? modeling weaknesses in unpersuasive arguments. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4082–4088.

Samuel L. Popkin. 1991. *The reasoning voter : communication and persuasion in presidential campaigns*. University of Chicago Press.

Peter Potash and Anna Rumshisky. 2017. Towards debate automation: a recurrent model for predicting debate winners. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2475, Copenhagen, Denmark, September. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Comput. Linguist.*, 43(3):619–659, September.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning Arguments. In *Proceedings of the 25th International Conference on World Wide Web - WWW '16*, pages 613–624, New York, New York, USA. ACM Press.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200, Berlin, Germany, August. Association for Computational Linguistics.

Lu Xiao and Taraneh Khazaei. 2019. Changing Others' Beliefs Online. In *Proceedings of the 10th International Conference on Social Media and Society - SMSociety '19*, pages 92–101, New York, New York, USA. ACM Press.

Diyi Yang, Jiaao Chen, Zichao Yang, Dan Jurafsky, and Eduard Hovy. 2019. Let's make your request more persuasive: Modeling persuasive strategies via semi-supervised neural nets on crowdfunding platforms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3620–3630.

Xiao Zhang and Dan Goldwasser. 2019. Sentiment tagging with partial labels using modular architectures. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 579–590.