Competing Independent Modules for Knowledge Integration and Optimization

Parsa Bagherzadeh and Sabine Bergler
CLaC Labs, Concordia University
Montréal, Canada
{p_bagher, bergler}@cse.concordia.ca

Abstract

This paper presents a neural framework of untied independent modules, used here for integrating off the shelf knowledge sources such as language models, lexica, POS information, and dependency relations. Each knowledge source is implemented as an independent component that can interact and share information with other knowledge sources. We report proof of concept experiments for several standard sentiment analysis tasks and show that the knowledge sources interoperate effectively without interference. As a second use-case, we show that the proposed framework is suitable for optimizing BERT-like language models even without the help of external knowledge sources. We cast each Transformer layer as a separate module and demonstrate performance improvements from this explicit integration of the different information encoded at the different Transformer layers .

1 Introduction

Pre-trained language models such as BERT (Devlin et al., 2019) are trained on large corpora with unsupervised end-to-end training. Such monolithic systems cannot take advantage of extant outside grammatical or domain knowledge as complementary information.

Many language tasks benefit from knowledge sources that are known a priori. For sentiment analysis, for instance, various gazetteer lists and sentiment lexica encode sentiment words, word polarity, aspect-sentiment pairs, etc., which were proven to be effective as knowledge sources in different machine learning architectures (Özdemir and Bergler, 2015; Yang et al., 2019; Zhao et al., 2020; Ke et al., 2020).

Deep learning systems for sentiment analysis leverage sentiment words to enhance embedding representations by continuing the pre-training process of masked language models (Tian et al., 2020), or by re-training a modified version of language model that has intermediate layers for explicit encoding of sentiment knowledge (Ke et al., 2020). This knowledge integration into pre-trained models exceeds fine-tuning in computational cost and requires sophisticated training phase calibration. Moreover, reported approaches only encode a single type of knowledge, either lexical (Tian et al., 2020) or grammatical (Tang et al., 2020). For tasks that benefit from several types of knowledge, repeated retraining becomes prohibitive.

This paper demonstrates the feasibility of using standard pre-trained language models and incorporate external off the shelf knowledge sources through dedicated and independent modules for each knowledge component. This framework reduces to some degree the need of ML experts for feature engineering for specific tasks, as creation of gazetteer lists and specialist lexica with domain scores is accessible to domain experts.

Modules incorporating knowledge sources are independent of one another to address major issues with combining extant knowledge with monolithic architectures, robustness, flexibility, and transparency.

Robustness Independent modules make it possible to exploit multiple, possibly inconsistent knowledge sources in parallel while reducing interference effects. Different knowledge sources that do not always agree on facts can cover a wider spectrum for the task at hand, but the inconsistency might hurt the leaning process. By spreading backpropagation independently over each module, training will weigh and assess usefulness of each module in context of the task and other modules. Schölkopf et al. (2012), Goyal et al. (2019) have shown that independent modules make the overall system more robust in case of distribution shifts.

Flexibility Since the modules are independent of one another, a module can be added/removed without further adjustments and, as our ablation experiments show, with only commensurate loss. Moreover, the independent modules can be deployed at different points in the architecture. This paper demonstrates flexibility of the approach by showing effectiveness for input oriented modules that are concerned with token level information (i.e. sentiment value or POS tag), with relation information between tokens (grammatical dependencies), and modules at the Transformer layer level, taking each Transformer layer as an independent source of information and improving performance significantly. At the Transformer layer level, modules permit the state of higher Transformer layers to influence the state of lower layers at low cost.

Transparency Module activation can be tracked. Because the modules here encode different components independently, their activation patterns can be visualized and analyzed, an important advantage, especially for domain expert developers who might not be familiar with development environments used by ML experts.

This paper presents a proof of concept of the interacting independent module framework on various sentiment analysis datasets. Sentiment analysis is a much studied topic with different general sentiment lexica readily available and well understood benefits from dependency parses and domain specific sentiment lexica. The sentiment tasks we use span a range from simple analysis of a datapoint as two class classification task, as a three class classification, as a relationship oriented aspect based sentiment classification, and finally sentiment classification for tweets expressing figurative language. This variety of task structures for which we can use the same knowledge sources makes the results comparable and showcases the flexibility and robustness of the modules.

2 Related literature

Neural modular design has been the topic of interest for more than three decades (Bottou and Gallinari, 1991), (Jacobs et al., 1991), (Ronco et al., 1996), (Reed and De Freitas, 2016). Most models proposed assume that only one expert is active at a particular time step but EntNet (Henaff et al., 2017) and IndRNN (Li et al., 2018) are propoals for sets of separate recurrent models, offering module independence, but no communication between modules. The recently proposed Recurrent Independent Mechanisms (RIMs) (Goyal et al., 2019), however, suggest to model a complex system by dividing the overall model into M communicative recurrent modules. The RIMs architecture was introduced for visual input. The independent mechanisms operate on the same input and do not have access to external information but rather make each module to specialize on a simpler problem by focusing in different parts of input.

Attempts at importing outside knowledge into neural architectures for language tasks have experimented with stacking (bi-)LSTMs (Søgaard and Goldberg, 2016), where we could interpret each layer of (bi-)LSTMs as a different module but with no independence and only one-way communication. For transformer architectures, adapters form a kind of module (Houlsby et al., 2019; Pfeiffer et al., 2020). Adapters are trainable modules and can be interspersed between attention layers of frozen language models to provide a *boost* by learning task specific representations.

The current proposal draws on several of these previous systems for a comprehensive architecture, where the independent modules can take different encodings as input.

3 The proposed framework

Token level knowledge sources Suppose there are $\mathcal{N}$ knowledge sources ($n = 1, \ldots, \mathcal{N}$) available. The annotations provided by nth knowledge source is encoded by an embedder E_n. Formally, E_n produces a sequence $\langle x_1^n, x_2^n, \ldots, x_T^n \rangle$ such as a token embeddings sequence, gazetteer lookup sequence, etc.)

Recurrent modules The output sequence of each embedder E_n, is used as input to a recurrent module R_n ($n = 1, \ldots, \mathcal{N}$). The modules are independent in their dynamics and can be chosen independently of any recurrent model, such as simple RNNs (Elman, 1990), GRUs (Cho et al., 2014), LSTMs (Hochreiter and Schmidhuber, 1997), Graph LSTMs (Peng et al., 2017), etc. We associate two hidden states with each module R_n at time-step t, a temporary hidden state $\tilde{h}_t^n \in \mathbb{R}^{d_h}$ and an actual hidden state $h_t^n \in \mathbb{R}^{d_h}$.

Controller A controller component $\mathcal{C}$, in this paper a LSTM, schedules read operations. At time-step t, the controller has the hidden state $z_t \in \mathbb{R}^{d_{cont}}$ and attends to the hidden states of all modules at $t-1$ and to position t on all of $\mathcal{N}$

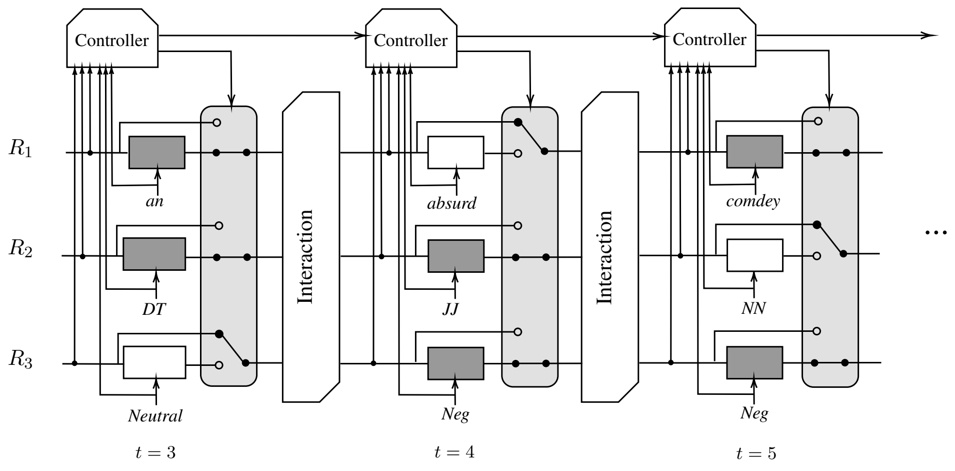

Figure 1: An illustration of the proposed framework with 3 recurrent modules (R_1-R_3) and $k = 2$. Each module processes a knowledge source; *Language model* $\rightarrow R_1$, *POS* $\rightarrow R_2$, and *Sentiment lexicon* $\rightarrow R_3$

input sequences:

$$z_t = \mathcal{C}(B_t, z_{t-1}) \tag{1}$$

where z_{t-1} is the previous hidden state of the controller and

$$B_t = softmax\left(\frac{Q_{t-1}(K_t)^T}{\sqrt{d_h}}\right) V_t \tag{2}$$

where $Q_{t-1} = z_{t-1}W^{query}$ and

$$K_t = [A_t^1 W_1^{key} \oplus \ldots \oplus A_t^{\mathcal{N}} W_{\mathcal{N}}^{key}]$$

$$V_t = [A_t^1 W_1^{val} \oplus \ldots \oplus A_t^{\mathcal{N}} W_{\mathcal{N}}^{val}]$$

$$A_t^n = [h_{t-1}^n; x_t^n]$$

where $W^{query} \in \mathbb{R}^{d^{cont} \times d^{query}}$ is the linear transformation for constructing query and $W_n^{key} \in \mathbb{R}^{(d_h+d_{in}) \times d^{key}}$ and $W_n^{val} \in \mathbb{R}^{(d_h+d_{in}) \times d^{val}}$ are linear transformations for constructing keys and values in the attention mechanism (Vaswani et al., 2017).

Note that the hidden state of the controller, z_t, is used to construct the query Q_t, to attend to the input sequences at the next time-step. In Equation 2 the $softmax$ produces $\mathcal{N}$ attention scores, each corresponding to a module. The top k modules form a subset S_t of recurrent modules that are active and thus will be updated by their respective input. Inactive modules output their input unchanged.

Updating recurrent modules All active modules produce a temporary hidden state:

$$\tilde{h}_t^n = R_k(x_t^n, h_{t-1}^n) \quad \forall n \in S_t \tag{3}$$

Interaction The module R_n attends attends to all other modules :

$$h_t^n = softmax\left(\frac{\tilde{Q}_t^n(\tilde{K}_t^{1:\mathcal{N}})^T}{\sqrt{d_h}}\right) \tilde{V}_t^{1:\mathcal{N}} \tag{4}$$

where $\tilde{Q}_t^n = \tilde{h}_t^n \tilde{W}_n^{query}$, $\tilde{W}_n^{query} \in \mathbb{R}^{d_h \times d_{int}^{query}}$ and

$$\tilde{K}_t^{1:\mathcal{N}} = [\tilde{h}_t^1 \tilde{W}_1^{key} \oplus \ldots \oplus \tilde{h}_t^{\mathcal{N}} \tilde{W}_{\mathcal{N}}^{key}]$$

$$\tilde{V}_t^{1:\mathcal{N}} = [\tilde{h}_t^1 \tilde{W}_1^{val} \oplus \ldots \oplus \tilde{h}_t^{\mathcal{N}} \tilde{W}_{\mathcal{N}}^{val}]$$

where $\tilde{W}_n^{key} \in \mathbb{R}^{d_h \times d_{int}^{key}}$ and $\tilde{W}_n^{val} \in \mathbb{R}^{d_h \times d_{int}^{val}}$ are linear transformations for constructing key and value for the interaction attention mechanism (Eq. 4) respectively.

An illustration of the proposed model is provided in Figure 1. Active modules are indicated in dark gray. At each time-step the controller determines the set of top k active modules (in the figure $k = 2$) by attending to inputs as well as all hidden states.

4 Tasks and datasets

We use different sentiment tasks and datasets to ensure that our observations are not task specific. We use only sentiment tasks, so that the same or similar external knowledge sources will be effective, again to ensure the observations are not specific to a (type of) knowledge source. In particular, we use

SST-2 Stanford sentiment tree-bank for binary sentiment classification of movie reviews (Socher et al., 2013), GLUE benchmark version[1] (Wang et al., 2018)

[1]http://gluebenchmark.com

SE17-4A SemEval 2017 task 4 subtask A for 3-class sentiment classification of tweets into *negative*, *neutral* and *positive* classes (Rosenthal et al., 2017)

SE14-5L SemEval 2015 task 5 for aspect-based sentiment analysis of online reviews of laptops. SE15-5L is a relation extraction and classification task.

SE14-5R SemEval 2015 task 5 for aspect-based sentiment analysis of online reviews of restaurants SE15-5R is a relation extraction and classification task.

SE15-11 SemEval 2015 task 11 for sentiment analysis of tweets expressing figurative language, including sarcastic, ironic, and metaphoric tweets (Ghosh et al., 2015)

5 Experiments

Our experiments divide into two sets: first, we show through ablation that importing external knowledge with interacting independent modules is effective for all tasks and that the modules do not interfere with each other. A second set of experiments makes each Transformer layer in two BERT-like models an independent interacting module and shows improved performance.

5.1 External knowledge sources

The most widely used external knowledge stems from pre-trained word embeddings. All experimental runs contain a module for token embeddings. We compare BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).

Task oriented knowledge sources for sentiment tasks include six sentiment lexica. We use three general sentiment lexica, which range from small to very big and from manually to automatically derived (AFINN (Nielsen, 2011), MPQA (Wilson et al., 2005), NRC HashTag Sentiment (Mohammad et al., 2013)) as well as NRC EmoLex (Mohammad and Turney, 2013). We also use two aspect specific lexica for the restaurant and laptop domain (Yelp (Kiritchenko et al., 2014), LapTop (Kiritchenko et al., 2014)).

Part-of-speech (POS) tags are the most widely used grammatical feature and are available from many standard NLP environments. We use ANNIE for POS tagging (Cunningham et al., 2002).

Dong et al. (2014); Huang and Carley (2019); Zhang et al. (2019); Veyseh et al. (2020) demonstrate the efficacy of dependency relations especially for aspect-based sentiment analysis. We use the Stanford Parser (Klein and Manning, 2003; de Marneffe et al., 2006) for extracting dependency relations.[2]

5.1.1 Implementation of modules

Here, an embedder E_n is either a pre-trained language model or a learnable embedding layer. For the token at position t, E_n emits its knowledge representation $x_t^n \in \mathbb{R}^{d_{in}^n}$, which is then used as input to the recurrent layer R_n which can be a LSTM, GRU, simple RNN, etc.

1. Token: The embeddings provided by the last layer of BERT$_{Base}$ or RoBERTa$_{Base}$ (layer 12) are used to embed tokens and form the input to a bi-LSTM.

2. POS: Following (Bagherzadeh and Bergler, 2021), we use Word2Vec to obtain a set of pre-trained vectors for POS tags.[3] The resulting POS embeddings form the input to a bi-LSTM module.

3. Sentiment: The AFINN, NRC HashTag Sentiment, Yelp, and Laptop lexica return sentiment scores or polarities numerically and can be directly used as input for their dedicated modules. The MPQA polarities $Negative$, $Neutral$, and $Positive$ are encoded as -1, 0, and $+1$. NRC EmoLex returns eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive), which directly maps into a n-hot vector representation. All sentiment score representations form input to bi-RNNs.

4. Dependencies: We use the Graph-LSTM model proposed by (Peng et al., 2017) to encode dependency information, because, its recurrent dynamics easily fit into the framework.

   The Graph-LSTM model encodes dependency relations using a bi-directional recurrent architecture, where the forward pass encodes all of the dependencies from a dependency

[2]We preprocess the data using a GATE pipeline (Cunningham et al., 2002) with the ANNIE English Tokenizer (for SST-2 and SE14) and ANNIE tweet tokenizer as well as the hashtag tokenizer for tweets.

[3]The Word2Vec model is trained on a combined set of all tasks

parse where the dependent follows the governor, and the backward pass encodes those dependencies, where the dependent precedes the governor in the input sequence (see Figure 2). At time-step t the input to the recurrent module is the token at position t as well as the hidden states at all time points corresponding to its governors.

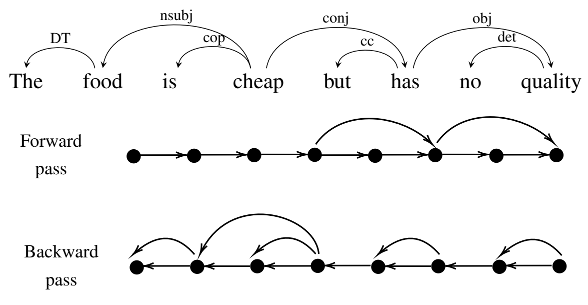

Figure 2: Graph-LSTM model for encoding dependency relations

Note that for high-dimensional and complex inputs such as word embeddings and POS embeddings, we used LSTMs. In order to keep the model light-weight, however, we used simple RNNs for the simpler encodings of sentiment lexica.

The models are implemented using PyTorch (Paszke et al., 2017). To calculate classification loss we use cross-entropy loss and we optimize the models using the Adam optimizer (Kingma and Ba, 2015) and the set of hyper-parameters is provided in Figure 3.

Module	d_{in}	d_h	d^{query}	d^{key}	d^{val}	d^{query}_{int}	d^{key}_{int}	d^{val}_{int}
Token	768	256	128	128	256	64	64	256
POS	50	256	128	128	256	64	64	256
Dep	100	256	128	128	256	64	64	256
Senti[1]	1	256	128	128	256	64	64	256
EmoLex	6	256	128	128	256	64	64	256

1: AFIIN, MPQA, NRC HashTag, Yelp, LapTop

Figure 3: Hyper-parameters used in the experiments

5.1.2 Results

We conduct ablation experiments for the different knowledge modules over all tasks, distinguishing the cases when all of the modules are active (Figure 4) and when only the top k modules are active (Figure 5). To highlight the potential of modules to compensate for the loss of fine tuning, we report the performance for frozen language models.

Baselines We report the performance of BERT and RoBERTa with no additional modules for comparison. Within our framework, the baseline case is when *Token* embeddings are the only input ($\mathcal{N} = 1$) and the model is reduced to a simple bi-LSTM over BERT or RoBERTa embeddings as input. Figure 4 suggests that this baseline is at least equivalent to the fine-tuned language models, as it never underperforms them.

All modules active Figure 4 shows that all runs consistently benefit from each of the sentiment lexica. Moreover, the difference between the different lexica is consistently surprisingly small. As expected, *HashTag Sent.* lexicon shows slightly greater improvements for the tweet data sets SE17-4 and SE15-11[4] and that the two specialized lexica *LapTop* and *Yelp* show slightly greater improvements in SE14-L and SE14-R. This pattern suggests that the system has properly attended to the different lexica.

Grammatical knowledge from POS and dependency relations also provides greater improvements for the aspect-based tasks, confirming the importance of grammatical knowledge for relation extraction. While both grammatical features yield only marginal improvements, their combination yields consistently better results, more notably for BERT-based settings and most significantly for the relation tasks SE14-L and SE14-R. Moreover, the two grammatical knowledge sources never lower performance.

A consistent observation among all settings is that the modules combine without loss in performance, and best results are consistently achieved when all modules are implemented ($\mathcal{N} = 9$). Note that when all modules are active, no controller component is needed.

Figure 4 also shows results for the frozen language models. For all five tasks and both language models, the full set of nine models fully compensates for fine tuning and even slightly increases performance above the fine-tuned baseline. Freezing language models can prevent over-fitting on small data sets. When language models are frozen in the Adapter framework (Pfeiffer et al., 2020), the Adapter modules become responsible for learning the inductive bias. In our framework in contrast, learning is facilitated by extant domain knowledge at a much reduced parameter space. On average, when the language models are frozen, the proposed

[4] *HashTag Sent.* is complied from tweet corpora.

			Acc		mac-Rec		mac-F1		mac-F1		Cosine	
			SST-2		SE17-4		SE14-L		SE14-R		SE15-11	
Fine-tune	Annotations	$\mathcal{N}$	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
✓	No modules	–	91.8	94.6	66.9	69.8	69.3	72.1	72.0	75.5	78.1	81.7
✓	Token	1	92.2	94.8	67.2	70.2	69.7	72.6	72.3	75.6	79.1	82.2
✓	Token, AFINN	2	92.8	95.3	69.1	71.6	71.8	72.9	72.9	75.8	80.3	83.0
✓	Token, MPQA	2	92.4	95.1	68.2	71.2	71.7	72.8	73.0	75.1	80.0	83.2
✓	Token, HashTag Sent	2	92.3	94.9	69.7	71.5	70.4	72.6	72.8	75.9	80.9	83.9
✓	Token, EmoLex	2	92.6	95.2	69.1	72.3	70.2	72.9	73.2	76.2	80.6	83.5
✓	Token, LapTop	2	92.3	94.8	67.8	70.2	72.0	73.9	72.5	75.8	79.5	82.4
✓	Token, Yelp	2	92.3	94.9	67.5	70.2	69.9	72.7	74.7	76.6	79.5	82.6
✓	Token, POS	2	92.4	95.0	68.2	70.2	70.6	73.0	74.3	76.8	79.6	82.2
✓	Token, Dep	2	92.7	95.0	67.6	70.8	72.8	74.6	76.8	78.4	79.8	82.7
✓	Token, POS, Dep	3	92.9	95.1	68.4	70.8	73.0	74.6	76.8	78.5	80.1	82.7
✓	Token, All Sent	7	94.4	95.7	71.2	73.4	74.6	75.8	75.9	78.7	83.3	85.1
✓	Token, All Sent, POS	8	94.5	95.7	71.6	**73.5**	74.9	76.1	76.3	79.0	83.7	**85.2**
✓	Token, All Sent, POS, Dep	9	**94.9**	**95.6**	**71.7**	**73.5**	**75.4**	**77.2**	**78.8**	**80.1**	**83.8**	**85.2**
×	Token	1	88.6	90.2	62.1	65.5	65.2	69.2	69.4	73.6	74.1	77.1
×	Token, All Sent	2	90.1	92.7	65.1	69.6	69.8	72.9	72.8	75.3	78.8	82.1
×	Token, POS	2	89.7	91.6	63.4	68.0	67.3	71.7	69.8	74.0	76.3	78.5
×	Token, Dep	2	90.0	91.9	63.8	69.0	69.5	72.0	71.5	76.3	77.1	79.3
×	Token, All Sent, POS, Dep	9	**92.9**	**93.4**	**67.0**	**70.5**	**71.3**	**73.4**	**74.1**	**78.7**	**79.4**	**83.6**

Figure 4: Integration of external knowledge in $\mathcal{N}$ independent module. All of the modules are kept active ($k = \mathcal{N}$) and the last layer in BERT or RoBERTa is used to embed tokens.

			Acc		mac-Rec		mac-F1		mac-F1		Cosine	
			SST-2		SE17-4		SE14-R		SE14-L		SE15-11	
Fine-tune	$\mathcal{N}$	k	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
✓	9	6	95.2	**96.9**	72.0	73.6	75.7	77.6	78.5	79.4	84.3	85.5
✓	9	7	**95.3**	96.7	**72.3**	**74.0**	75.9	**77.8**	78.6	80.0	84.6	**85.6**
✓	9	8	95.0	96.2	71.9	73.8	75.8	77.4	79.0	**80.4**	83.9	85.2
✓	9	9	94.9	95.6	71.7	73.5	75.4	77.2	78.8	80.1	83.8	85.2
×	9	6	93.3	94.6	67.3	71.0	72.1	**74.6**	**75.6**	**79.6**	79.9	**84.2**
×	9	7	**93.5**	**94.8**	**67.8**	**71.3**	**72.4**	**74.6**	75.4	79.2	**80.0**	83.9
×	9	8	93.2	94.5	67.4	70.8	71.8	74.1	74.8	78.9	79.6	83.7
×	9	9	92.9	93.4	67.0	70.5	71.3	73.4	74.1	78.7	79.4	83.6

Figure 5: Integration of external knowledge in 9 independent modules (Token, All Sent, POS, Dep). Some of the modules are active ($k \leq \mathcal{N}$)

model has $15M$ trainable parameters. Reducing the inference and back-propagation timing from 1.8sec to 0.9sec.

Top k modules active When the set of active modules is limited to the top k, the modules compete with each other for active status (Figure 5). Interestingly, for all tasks, limiting the number of modules yields better performance and confirms the importance of sparse activation of the modules. For fine-tuned models, the best performer varies between 6, 7, and 8 active models. The differences are very small and thus merely suggestive. Interestingly, however, for the frozen language models, $k = 7$ is the most frequent best performer with the restaurant domain being an exception. The Figurative language task SE15-11 shows the only task for which BERT and RoBERTa frozen models differ in this respect.

Allowing only a set of top modules to be active resembles hard attention with two major differences: there is no need to apply a threshold value to the attention scores here (the threshold is the number of active modules k) and activity/inactivity is determined based on competition among modules in our framework.

Competition between modules fosters independence among learned mechanisms, making each module specialize on a simpler aspect of the problem (Goyal et al., 2019; Parascandolo et al., 2018). Here, we demonstrate system behavior by varying the number of active modules (k) manually. The k values for best-preforming settings fall within a narrow interval, suggesting that automatic mechanisms can determine k during training.

5.2 Integrating Transformer layers

The smallest version of BERT consists of 12 layers of Transformer encoders. Jawahar et al. (2019); Tenney et al. (2019); de Vries et al. (2020) all argued that layers in BERT-style models encode different information. For instance, (Jawahar et al., 2019) claim that phrase-level information is encoded in lower layers of BERT and intermediate layers encode linguistic information, with *surface features at the bottom and syntactic features in the middle*. Tenney et al. (2019) also demonstrate that lower layers in BERT provide richer representation

			Acc		mac-Rec		mac-F1		mac-F1		Cosine	
			SST-2		SE17-4		SE14-R		SE14-L		SE15-11	
Fine-tune	Layers	$\mathcal{N}$	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
✓	12	1	92.2	94.8	67.2	70.2	69.7	72.6	72.3	75.6	79.1	82.2
	1–12	12	93.1	95.1	70.2	71.6	73.1	73.5	73.7	77.0	82.0	85.3
×	12	1	88.6	90.2	62.1	65.5	65.2	69.2	69.4	70.0	74.1	77.1
	1–12	12	91.9	92.5	68.4	68.7	69.1	71.8	71.9	73.6	78.6	80.1

Figure 6: Integration of $\mathcal{N}$ Transformer layers in $\mathcal{N}$ active independent modules

			Acc		mac-Rec		mac-F1		mac-F1		Cosine	
			SST-2		SE17-4		SE14-R		SE14-L		SE15-11	
Fine-tune	$\mathcal{N}$	k	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
		1	92.8	93.7	69.4	70.8	72.7	72.9	73.1	76.3	81.3	83.7
✓	12	6	**94.1**	**95.0**	**70.9**	**72.0**	**74.2**	**74.4**	**75.3**	**78.0**	**83.8**	**85.1**
		12	93.1	95.1	70.2	71.6	73.1	73.5	73.7	77.0	82.0	85.3
		1	89.7	91.0	66.7	67.0	67.3	69.5	70.3	71.2	75.9	79.3
×	12	6	**93.2**	**93.1**	**68.8**	**69.5**	**71.3**	**72.5**	**73.4**	**75.4**	**80.2**	**81.8**
		12	91.9	92.5	68.4	68.7	69.1	71.8	71.9	73.6	78.6	80.1

Figure 7: Integration of 12 Transformer layers (1-12) in k active independent modules

for POS tagging, concluding that the low-level layers implicitly encode POS information. For most tasks however, only the last layer of BERT-like language models is used to make predictions.

We demonstrate that the framework for independent, interacting modules, while useful for incorporating external knowledge sources into a neural architecture, is more generally beneficial. We encode each layer of two BERT-like language models in separate modules, thus enabling lower layers to have access to the representations of higher layers (bi-directional flow of information) and vice versa. We hypothesize that the framework can effectively combine all layers and yield improvements especially for tasks where different levels of representations are required, such as relation detection.

5.2.1 Results

Figure 6 show performances across the tasks, when the representations provided by Transformer layers are integrated in our independent modules. Integrating all 12 layers yields consistent improvements across all tasks when compared to the output of layer 12. This demonstrates the potential of bi-directional flow of information and the self-awareness of all intermediate layers.

Figure 7 shows results for running all 12 modules, but limiting activity to the top k. Only three different values for k are shown, 1, 6, and 12. Consistently, for all tasks and for both, fine-tuned and frozen models, $k = 6$ shows top performance, confirming the previous observation that competition increases performance.

The difference for the first rows in Figures 6 and 7 is instructive: The first row in Figure 6 shows performance of one single module with input from Transformer layer 12, while Figure 7 shows 12 modules with a limit of $k = 1$. Almost always, the 12 modules that select a top k find a slight improvement, which must be due to the interaction: while non-active modules do not update a hidden state, their hidden state is the hidden state of the previous time step. The active module can inspect these hidden states and thus potentially gain information.

To gain an overview over all layers and for all tasks, Figure 8 shows the percentage of time-steps where the independent modules for different Transformer layers have been active. For all tasks, the last layer is most active. This is not surprising since this layer is the target when pre-training language models. Interestingly, the first two layers also consistently demonstrate high activity, for all tasks. We surmise that this may be due to the fact that the sentiment tasks are token-oriented and the first two layers might capture lexical-triggers. The pronounced spike in activity for layer 7 for the aspect based tasks SE14-R and SE14-L might, likewise, confirm the conjecture by Jawahar et al. (2019) that intermediate layers encode grammatical relations, here possibly dependency relations.

6 Discussion

The reported experiments demonstrate the capabilities of the competing independent modules both for leveraging external knowledge and integrating Transformer layers of BERT-like models. In both cases, the integration leads to improvements.

Integration of layers is a strong competitor for independent modules that leverage external knowledge. This suggests that the required knowledge is already encoded in the language models to some extent. The question is: which approach is preferred?

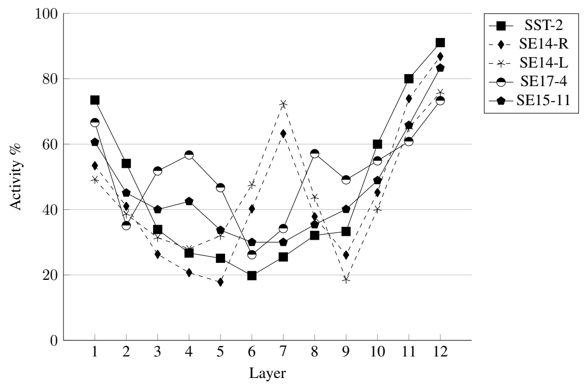

Figure 8: Percentage of time-steps that the independent modules have been active

The embedding dimensions for external knowledge sources such as sentiment lexcia and POS are very small. This leads to a small number of trainable parameters in the independent modules that encode these knowledge sources. When integrating layers, since the input dimensions for all independent module is the same (768 in our experiments), the number of trainable parameters is significantly larger compared to the case of external knowledge sources. The choice between the two options depends on the target task and the availability of task specific knowledge sources. When these resources are available, the reduction in development and processing effort becomes very attractive, especially for small datasets.

7 Conclusions

This paper presents a proof of concept for integrating external knowledge in competing, interacting independent modules. The reported experiments show consistent improvements when using readily available, off-the-shelf knowledge sources such as sentiment lexica, POS, and dependency relations encoded in independent modules. This is true even for knowledge sources that contradict each others' information, showing the robustness of the approach. When modules are in competition mode, further improvements can be achieved.

Experiments with two frozen language models demonstrate that task-specific knowledge sources in this architecture more than compensate for fine-tuning of the language model, with a significant reduction in the number of trainable parameters.

We also show that the proposed framework is suitable for integration of the Transformer layers of Transformer-based language models by allowing lower layers to have access to the representations of high layers, i.e. bottom-up and top-down flow of information.

Moreover, the behavior of the independent modules can be visualized and the contribution of each module can be measured.

In summary, interacting independent modules are a framework that enables computation restrained task adaptation with off-the-shelf external resources in a transparent fashion.

References

Parsa Bagherzadeh and Sabine Bergler. 2021. Leveraging knowledge sources for detecting self-reports of particular health issues on social media. In *Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis*, pages 38–48, online.

Léon Bottou and Patrick Gallinari. 1991. A framework for the cooperation of learning algorithms. In *Advances in neural information processing systems*, pages 781–788.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Wietse de Vries, Andreas van Cranenburgh, and Malvina Nissim. 2020. What's so special about BERT's layers? a closer look at the NLP pipeline in monolingual and multilingual models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4339–4350.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. SemEval 2015 Task 11: Sentiment analysis of figurative language in Twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 470–478.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. 2019. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR'17*.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Binxuan Huang and Kathleen M Carley. 2019. Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5472–5480.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657.

Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. 2020. SentiLARE: Sentiment-aware language representation learning with linguistic knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6975–6988.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR'15*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *ACL 2003*.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. 2018. Independently recurrent neural network (IndRNN): Building a longer and deeper RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational intelligence*, 29(3):436–465.

Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Workshop on 'Making Sense of Microposts: Big things come in small packages'*, pages 93–98.

Canberk Özdemir and Sabine Bergler. 2015. A comparative study of different sentiment lexica for sentiment analysis of tweets. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2015)*.

Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. 2018. Learning independent causal mechanisms. In *International Conference on Machine Learning (ICML)*, pages 4036–4044.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems 2017*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online.

Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. In *ICLR*.

Eric Ronco, Henrik Gollee, and Peter J Gawthrop. 1996. Modular neural network and self-decomposition. *Connection Science (special issue: COMBINING NEURAL NETS).(To appear)*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 Task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. 2012. On causal and anticausal learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, pages 1255–1262.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235.

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6578–6588.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, and Feng Wu. 2020. SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4067–4076.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Amir Pouran Ben Veyseh, Nasim Nouri, Franck Dernoncourt, Quan Hung Tran, Dejing Dou, and Thien Huu Nguyen. 2020. Improving aspect-based sentiment analysis with gated graph convolutional networks and syntax-based regulation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4543–4548.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4568–4578.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020. Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390.