Open-Domain Targeted Sentiment Analysis via Span-Based Extraction and Classification

Minghao Hu[†], **Yuxing Peng**[†], **Zhen Huang**[†], **Dongsheng Li**[†], **Yiwei Lv**[§]
[†] National University of Defense Technology, Changsha, China
[§] University of Macau, Macau, China
{huminghao09,pengyuxing,huangzhen,dsli}@nudt.edu.cn

Abstract

Open-domain targeted sentiment analysis aims to detect opinion targets along with their sentiment polarities from a sentence. Prior work typically formulates this task as a sequence tagging problem. However, such formulation suffers from problems such as huge search space and sentiment inconsistency. To address these problems, we propose a span-based extract-then-classify framework, where multiple opinion targets are directly extracted from the sentence under the supervision of target span boundaries, and corresponding polarities are then classified using their span representations. We further investigate three approaches under this framework, namely the pipeline, joint, and collapsed models. Experiments on three benchmark datasets show that our approach consistently outperforms the sequence tagging baseline. Moreover, we find that the pipeline model achieves the best performance compared with the other two models.

1 Introduction

Open-domain targeted sentiment analysis is a fundamental task in opinion mining and sentiment analysis (Pang et al., 2008; Liu, 2012). Compared to traditional sentence-level sentiment analysis tasks (Lin and He, 2009; Kim, 2014), the task requires detecting target entities mentioned in the sentence along with their sentiment polarities, thus being more challenging. Taking Figure 1 as an example, the goal is to first identify "*Windows 7*" and "*Vista*" as opinion targets and then predict their corresponding sentiment classes.

Sentence: I love **[Windows 7]**$_+$ which is a vast improvment over **[Vista]**..
Targets: Windows 7, Vista
Polarities: positive, negative

Figure 1: Open-domain targeted sentiment analysis.

Typically, the whole task can be decoupled into two subtasks. Since opinion targets are not given, we need to first detect the targets from the input text. This subtask, which is usually denoted as *target extraction*, can be solved by sequence tagging methods (Jakob and Gurevych, 2010; Liu et al., 2015; Wang et al., 2016a; Poria et al., 2016; Shu et al., 2017; He et al., 2017; Xu et al., 2018). Next, *polarity classification* aims to predict the sentiment polarities over the extracted target entities (Jiang et al., 2011; Dong et al., 2014; Tang et al., 2016a; Wang et al., 2016b; Chen et al., 2017; Xue and Li, 2018; Li et al., 2018; Fan et al., 2018). Although lots of efforts have been made to design sophisticated classifiers for this subtask, they all assume that the targets are already given.

Rather than using separate models for each subtask, some works attempt to solve the task in a more integrated way, by jointly extracting targets and predicting their sentiments (Mitchell et al., 2013; Zhang et al., 2015; Li et al., 2019). The key insight is to label each word with a set of target tags (e.g., B, I, O) as well as a set of polarity tags (e.g., +, -, 0), or use a more collapsed set of tags (e.g., B+, I-) to directly indicate the boundary of targeted sentiment, as shown in Figure 2(a). As a result, the entire task is formulated as a sequence tagging problem, and solved using either a pipeline model, a joint model, or a collapsed model under the same network architecture.

However, the above annotation scheme has several disadvantages in target extraction and polarity classification. Lee et al. (2016) show that, when using BIO tags for extractive question answering tasks, the model must consider a huge search space due to the compositionality of labels (the power set of all sentence words), thus being less effective. As for polarity classification, the sequence tagging scheme turns out to be problematic for two reasons. First, tagging polarity over each word

Sentence:	I	love	Windows	7	...	over	Vista	.
Pipeline/ Joint:	O	O	B	I		O	B	O
	0	0	+	+		0	-	0
Collapsed:	O	O	B+	I+		O	B-	O

(a) Sequence tagging. The B/I/O labels indicate target span boundaries, while +/-/0 refer to sentiment polarities.

Sentence:	I love Windows 7 ... over Vista .
Pipeline/ Joint:	Target start: 3, 11 Target end: 4, 11 Polarity: +, -
Collapsed:	Target start: 3+, 11- Target end: 4+, 11-

(b) Span-based labeling. The number denotes the start/end position of the given target in the sentence.

Figure 2: Comparison of different annotation schemes for the pipeline, joint, and collapsed models.

ignores the semantics of the entire opinion target. Second, since predicted polarities over target words may be different, the sentiment consistency of multi-word entity can not be guaranteed, as mentioned by Li et al. (2019). For example, there is a chance that the words "*Windows*" and "*7*" in Figure 2(a) are predicted to have different polarities due to word-level tagging decisions.

To address the problems, we propose a span-based labeling scheme for open-domain targeted sentiment analysis, as shown in Figure 2(b). The key insight is to annotate each opinion target with its span boundary followed by its sentiment polarity. Under such annotation, we introduce an extract-then-classify framework that first extracts multiple opinion targets using an heuristic multi-span decoding algorithm, and then classifies their polarities with corresponding summarized span representations. The advantage of this approach is that the extractive search space can be reduced linearly with the sentence length, which is far less than the tagging method. Moreover, since the polarity is decided using the targeted span representation, the model is able to take all target words into account before making predictions, thus naturally avoiding sentiment inconsistency.

We take BERT (Devlin et al., 2018) as the default backbone network, and explore two research questions. First, we make an elaborate comparison between tagging-based models and span-based models. Second, following previous works (Mitchell et al., 2013; Zhang et al., 2015), we compare the pipeline, joint, and collapsed models under the span-based labeling scheme. Extensive experiments on three benchmark datasets show that our models consistently outperform sequence tagging baselines. In addition, the pipeline model firmly improves over both the joint and collapsed models. Source code is released to facilitate future research in this field[1].

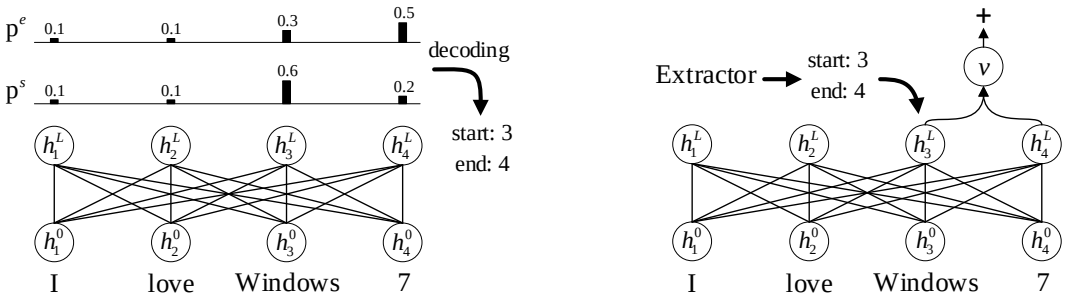
[1]https://github.com/huminghao16/SpanABSA

2 Related Work

Apart from sentence-level sentiment analysis (Lin and He, 2009; Kim, 2014), targeted sentiment analysis, which requires the detection of sentiments towards mentioned entities in the open domain, is also an important research topic.

As discussed in §1, this task is usually divided into two subtasks. The first is target extraction for identifying entities from the input sentence. Traditionally, Conditional Random Fields (CRF) (Lafferty et al., 2001) have been widely explored (Jakob and Gurevych, 2010; Wang et al., 2016a; Shu et al., 2017). Recently, many works concentrate on leveraging deep neural networks to tackle this task, e.g., using CNNs (Poria et al., 2016; Xu et al., 2018), RNNs (Liu et al., 2015; He et al., 2017), and so on. The second is polarity classification, assuming that the target entities are given. Recent works mainly focus on capturing the interaction between the target and the sentence, by utilizing various neural architectures such as LSTMs (Hochreiter and Schmidhuber, 1997; Tang et al., 2016a) with attention mechanism (Wang et al., 2016b; Li et al., 2018; Fan et al., 2018), CNNs (Xue and Li, 2018; Huang and Carley, 2018), and Memory Networks (Tang et al., 2016b; Chen et al., 2017; Li and Lam, 2017).

Rather than solving these two subtasks with separate models, a more practical approach is to directly predict the sentiment towards an entity along with discovering the entity itself. Specifically, Mitchell et al. (2013) formulate the whole task as a sequence tagging problem and propose to use CRF with hand-crafted linguistic features. Zhang et al. (2015) further leverage these linguistic features to enhance a neural CRF model. Recently, Li et al. (2019) have proposed a unified model that contains two stacked LSTMs along with carefully-designed components for maintaining sentiment consistency and improving target



(a) Multi-target extractor.

(b) Polarity classifier.

Figure 3: An overview of the proposed framework. Word embeddings are fed to the BERT encoder (Devlin et al., 2018) that contains L pre-trained Transformer blocks (Vaswani et al., 2017). The last block's hidden states are used to (a) propose one or multiple candidate targets based on the probabilities of the start and end positions, (b) predict the sentiment polarity using the span representation of the given target.

word detection. Our work differs from these approaches in that we formulate this task as a span-level extract-then-classify process instead.

The proposed span-based labeling scheme is inspired by recent advances in machine comprehension and question answering (Seo et al., 2017; Hu et al., 2018), where the task is to extract a continuous span of text from the document as the answer to the question (Rajpurkar et al., 2016). To solve this task, Lee et al. (2016) investigate several predicting strategies, such as BIO prediction, boundary prediction, and the results show that predicting the two endpoints of the answer is more beneficial than the tagging method. Wang and Jiang (2017) explore two answer prediction methods, namely the sequence method and the boundary method, finding that the later performs better. Our approach is related to this line of work. However, unlike these works that extract one span as the final answer, our approach is designed to dynamically output one or multiple opinion targets.

3 Extract-then-Classify Framework

Instead of formulating the open-domain targeted sentiment analysis task as a sequence tagging problem, we propose to use a span-based labeling scheme as follows: given an input sentence $\mathbf{x} = (x_1, ..., x_n)$ with length n, and a target list $\mathbf{T} = \{\mathbf{t}_1, ..., \mathbf{t}_m\}$, where the number of targets is m and each target $\mathbf{t}_i$ is annotated with its start position, its end position, and its sentiment polarity. The goal is to find all targets from the sentence as well as predict their polarities.

The overall illustration of the proposed framework is shown in Figure 3. The basis of our framework is the BERT encoder (Devlin et al., 2018): we map word embeddings into contextualized token representations using pre-trained Transformer blocks (Vaswani et al., 2017) (§3.1). A multi-target extractor is first used to propose multiple candidate targets from the sentence (§3.2). Then, a polarity classifier is designed to predict the sentiment towards each extracted candidate using its summarized span representation (§3.3). We further investigate three different approaches under this framework, namely the pipeline, joint, and collapsed models in §3.4.

3.1 BERT as Backbone Network

We use Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), a pre-trained bidirectional Transformer encoder that achieves state-of-the-art performances across a variety of NLP tasks, as our backbone network.

We first tokenize the sentence $\mathbf{x}$ using a 30,522 wordpiece vocabulary, and then generate the input sequence $\tilde{\mathbf{x}}$ by concatenating a `[CLS]` token, the tokenized sentence, and a `[SEP]` token. Then for each token $\tilde{x}_i$ in $\tilde{\mathbf{x}}$, we convert it into vector space by summing the token, segment, and position embeddings, thus yielding the input embeddings $\mathbf{h}^0 \in \mathbb{R}^{(n+2)\times h}$, where h is the hidden size.

Next, we use a series of L stacked Transformer blocks to project the input embeddings into a sequence of contextual vectors $\mathbf{h}^i \in \mathbb{R}^{(n+2)\times h}$ as:

$$\mathbf{h}^i = \text{TransformerBlock}(\mathbf{h}^{i-1}), \forall i \in [1, L]$$

Here, we omit an exhaustive description of the block architecture and refer readers to Vaswani et al. (2017) for more details.

3.2 Multi-Target Extractor

Multi-target extractor aims to propose multiple candidate opinion targets (Figure 3(a)). Rather than finding targets via sequence tagging methods, we detect candidate targets by predicting the start and end positions of the target in the sentence, as suggested in extractive question answering (Wang and Jiang, 2017; Seo et al., 2017; Hu et al., 2018). We obtain the unnormalized score as well as the probability distribution of the start position as:

$$\mathbf{g}^s = \mathbf{w}_s \mathbf{h}^L \ , \ \mathbf{p}^s = \mathrm{softmax}(\mathbf{g}^s)$$

where $\mathbf{w}_s \in \mathbb{R}^h$ is a trainable weight vector. Similarly, we can get the probability of the end position along with its confidence score by:

$$\mathbf{g}^e = \mathbf{w}_e \mathbf{h}^L \ , \ \mathbf{p}^e = \mathrm{softmax}(\mathbf{g}^e)$$

During training, since each sentence may contain multiple targets, we label the span boundaries for all target entities in the list $\mathbf{T}$. As a result, we can obtain a vector $\mathbf{y}^s \in \mathbb{R}^{(n+2)}$, where each element $\mathbf{y}^s_i$ indicates whether the i-th token starts a target, and also get another vector $\mathbf{y}^e \in \mathbb{R}^{(n+2)}$ for labeling the end positions. Then, we define the training objective as the sum of the negative log probabilities of the true start and end positions on two predicted probabilities as:

$$\mathcal{L} = -\sum_{i=1}^{n+2} \mathbf{y}^s_i \log(\mathbf{p}^s_i) - \sum_{j=1}^{n+2} \mathbf{y}^e_j \log(\mathbf{p}^e_j)$$

At inference time, previous works choose the span (k, l) $(k \leq l)$ with the maximum value of $\mathbf{g}^s_k + \mathbf{g}^e_l$ as the final prediction. However, such decoding method is not suitable for the multi-target extraction task. Moreover, simply taking top-K spans according to the addition of two scores is also not optimal, as multiple candidates may refer to the same text. Figure 4 gives a qualitative example to illustrate this phenomenon.

Sentence: Great food but the service was dreadful!
Targets: food, service
Predictions: food but the service, food, Great food, service, service was dreadful, ...

Figure 4: An example shows that there are many redundant spans in top-K predictions.

To adapt to multi-target scenarios, we propose an heuristic multi-span decoding algorithm as shown in Algorithm 1. For each example, top-M indices are first chosen from the two predicted scores $\mathbf{g}^s$ and $\mathbf{g}^e$ (line 2), and the candidate span $(\mathbf{s}_i, \mathbf{e}_j)$ (denoted as $\mathbf{r}_l$) along with its heuristic-regularized score $\mathbf{u}_l$ are then added to the lists $\mathbf{R}$ and $\mathbf{U}$ respectively, under the constraints that the end position is no less than the start position as well as the addition of two scores exceeds a threshold γ (line 3-8). Note that we heuristically calculate $\mathbf{u}_l$ as the sum of two scores minus the span length (line 6), which turns out to be critical to the performance as targets are usually short entities. Next, we prune redundant spans in $\mathbf{R}$ using the non-maximum suppression algorithm (Rosenfeld and Thurston, 1971). Specifically, we remove the span $\mathbf{r}_l$ that possesses the maximum score $\mathbf{u}_l$ from the set $\mathbf{R}$ and add it to the set $\mathbf{O}$ (line 10-11). We also delete any span $\mathbf{r}_k$ that is overlapped with $\mathbf{r}_l$, which is measured with the word-level F1 function (line 12-14). This process is repeated for remaining spans in $\mathbf{R}$, until $\mathbf{R}$ is empty or top-K target spans have been proposed (line 9).

Algorithm 1 Heuristic multi-span decoding

Input: $\mathbf{g}^s, \mathbf{g}^e, \gamma, K$
$\mathbf{g}^s$ denotes the score of start positions
$\mathbf{g}^e$ denotes the score of end positions
γ is a minimum score threshold
K is the maximum number of proposed targets

1: Initialize $\mathbf{R}, \mathbf{U}, \mathbf{O} = \{\}, \{\}, \{\}$
2: Get top-M indices $\mathbf{S}, \mathbf{E}$ from $\mathbf{g}^s, \mathbf{g}^e$
3: **for** $\mathbf{s}_i$ in $\mathbf{S}$ **do**
4: **for** $\mathbf{e}_j$ in $\mathbf{E}$ **do**
5: **if** $\mathbf{s}_i \leq \mathbf{e}_j$ and $\mathbf{g}^s_{\mathbf{s}_i} + \mathbf{g}^e_{\mathbf{e}_j} \geq \gamma$ **then**
6: $\mathbf{u}_l = \mathbf{g}^s_{\mathbf{s}_i} + \mathbf{g}^e_{\mathbf{e}_j} - (\mathbf{e}_j - \mathbf{s}_i + 1)$
7: $\mathbf{r}_l = (\mathbf{s}_i, \mathbf{e}_j)$
8: $\mathbf{R} = \mathbf{R} \cup \{\mathbf{r}_l\}, \mathbf{U} = \mathbf{U} \cup \{\mathbf{u}_l\}$
9: **while** $\mathbf{R} \neq \{\}$ and size$(\mathbf{O}) < K$ **do**
10: $l = \arg\max \mathbf{U}$
11: $\mathbf{O} = \mathbf{O} \cup \{\mathbf{r}_l\}; \mathbf{R} = \mathbf{R} - \{\mathbf{r}_l\}; \mathbf{U} = \mathbf{U} - \{\mathbf{u}_l\}$
12: **for** $\mathbf{r}_k$ in $\mathbf{R}$ **do**
13: **if** $\mathrm{f1}(\mathbf{r}_l, \mathbf{r}_k) \neq 0$ **then**
14: $\mathbf{R} = \mathbf{R} - \{\mathbf{r}_k\}; \mathbf{U} = \mathbf{U} - \{\mathbf{u}_k\}$
15: **return** $\mathbf{O}$

3.3 Polarity Classifier

Typically, polarity classification is solved using either sequence tagging methods or sophisticated neural networks that separately encode the target and the sentence. Instead, we propose to summarize the target representation from contextual sentence vectors according to its span boundary, and use feed-forward neural networks to predict the sentiment polarity, as shown in Figure 3(b).

Specifically, given a target span $\mathbf{r}$, we calculate a summarized vector $\mathbf{v}$ using the attention mechanism (Bahdanau et al., 2014) over tokens in its

corrsponding bound $(\mathbf{s}_i, \mathbf{e}_j)$, similar to Lee et al. (2017) and He et al. (2018):

$$\alpha = \text{softmax}(\mathbf{w}_\alpha \mathbf{h}^L_{\mathbf{s}_i:\mathbf{e}_j})$$
$$\mathbf{v} = \sum_{t=\mathbf{s}_i}^{\mathbf{e}_j} \alpha_{t-\mathbf{s}_i+1} \mathbf{h}^L_t$$

where $\mathbf{w}_\alpha \in \mathbb{R}^h$ is a trainable weight vector.

The polarity score is obtained by applying two linear transformations with a Tanh activation in between, and is normalized with the softmax function to output the polarity probability as:

$$\mathbf{g}^p = \mathbf{W}_p \text{tanh}(\mathbf{W}_v \mathbf{v}) \ , \ \mathbf{p}^p = \text{softmax}(\mathbf{g}^p)$$

where $\mathbf{W}_v \in \mathbb{R}^{h \times h}$ and $\mathbf{W}_p \in \mathbb{R}^{k \times h}$ are two trainable parameter matrices.

We minimize the negative log probabilities of the true polarity on the predicted probability as:

$$\mathcal{J} = -\sum_{i=1}^{k} \mathbf{y}^p_i \log(\mathbf{p}^p_i)$$

where $\mathbf{y}^p$ is an one-hot label indicating the true polarity, and k is the number of sentiment classes.

During inference, the polarity probability is calculated for each candidate target span in the set $\mathbf{O}$, and the sentiment class that possesses the maximum value in $\mathbf{p}^p$ is chosen.

3.4 Model Variants

Following Mitchell et al. (2013); Zhang et al. (2015), we investigate three kinds of models under the extract-then-classify framework:

Pipeline model We first build a multi-target extractor where a BERT encoder is exclusively used. Then, a second backbone network is used to provide contextual sentence vectors for the polarity classifier. Two models are separately trained and combined as a pipeline during inference.

Joint model In this model, each sentence is fed into a shared BERT backbone network that finally branches into two sibling output layers: one for proposing multiple candidate targets and another for predicting the sentiment polarity over each extracted target. A joint training loss $\mathcal{L} + \mathcal{J}$ is used to optimize the whole model. The inference procedure is the same as the pipeline model.

Collapsed model We combine target span boundaries and sentiment polarities into one label space. For example, the sentence in Figure 2(b) has a positive span $(3+, 4+)$ and a negative span $(11\text{-}, 11\text{-})$. We then modify the multi-target extractor by producing three sets of probabilities of the start and end positions, where each set corresponds to one sentiment class (e.g., $\mathbf{p}^{s+}$ and $\mathbf{p}^{e+}$ for positive targets). Then, we define three objectives to optimize towards each polarity. During inference, the heuristic multi-span decoding algorithm is performed on each set of scores (e.g., $\mathbf{g}^{s+}$ and $\mathbf{g}^{e+}$), and the output sets $\mathbf{O}^+$, $\mathbf{O}^-$, and $\mathbf{O}^0$ are aggregated as the final prediction.

Dataset	#Sent	#Targets	#+	#-	#0
LAPTOP	1,869	2,936	1,326	990	620
REST	3,900	6,603	4,134	1,538	931
TWITTER	2,350	3,243	703	274	2,266

Table 1: Dataset statistics. '#Sent' and '#Targets' denote the number of sentences and targets, respectively. '+', '-', and '0' refer to the positive, negative, and neutral sentiment classes.

4 Experiments

4.1 Setup

Datasets We conduct experiments on three benchmark sentiment analysis datasets, as shown in Table 1. LAPTOP contains product reviews from the laptop domain in SemEval 2014 ABSA challenges (Pontiki et al., 2014). REST is the union set of the restaurant domain from SemEval 2014, 2015 and 2016 (Pontiki et al., 2015, 2016). TWITTER is built by Mitchell et al. (2013), consisting of twitter posts. Following Zhang et al. (2015); Li et al. (2019), we report the ten-fold cross validation results for TWITTER, as there is no train-test split. For each dataset, the gold target span boundaries are available, and the targets are labeled with three sentiment polarities, namely *positive* (+), *negative* (-), and *neutral* (0).

Metrices We adopt the precision (P), recall (R), and F1 score as evaluation metrics. A predicted target is correct only if it exactly matches the gold target entity and the corresponding polarity. To separately analyze the performance of two subtasks, precision, recall, and F1 are also used for the target extraction subtask, while the accuracy (ACC) metric is applied to polarity classification.

Model settings We use the publicly available $\text{BERT}_{\text{LARGE}}$[2] model as our backbone network,

[2]https://github.com/google-research/bert

Model	LAPTOP			REST			TWITTER		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
UNIFIED	61.27	54.89	57.90	68.64	71.01	69.80	53.08	43.56	48.01
TAG-pipeline	65.84	67.19	66.51	71.66	76.45	73.98	54.24	54.37	54.26
TAG-joint	65.43	66.56	65.99	71.47	75.62	73.49	54.18	54.29	54.20
TAG-collapsed	63.71	66.83	65.23	71.05	75.84	73.35	54.05	54.25	54.12
SPAN-pipeline	69.46	66.72	**68.06**	76.14	73.74	**74.92**	60.72	55.02	**57.69**
SPAN-joint	67.41	61.99	64.59	72.32	72.61	72.47	57.03	52.69	54.55
SPAN-collapsed	50.08	47.32	48.66	63.63	53.04	57.85	51.89	45.05	48.11

Table 2: Main results on three benchmark datasets. A $\text{BERT}_{\text{LARGE}}$ backbone network is used for both the "TAG" and "SPAN" models. State-of-the-art results are marked in **bold**.

and refer readers to Devlin et al. (2018) for details on model sizes. We use Adam optimizer with a learning rate of 2e-5 and warmup over the first 10% steps to train for 3 epochs. The batch size is 32 and a dropout probability of 0.1 is used. The number of candidate spans M is set as 20 while the maximum number of proposed targets K is 10 (Algorithm 1). The threshold γ is manually tuned on each dataset. All experiments are conducted on a single NVIDIA P100 GPU card.

4.2 Baseline Methods

We compare the proposed span-based approach with the following methods:

TAG-{pipeline, joint, collapsed} are the sequence tagging baselines that involve a BERT encoder and a CRF decoder. "pipeline" and "joint" denote the pipeline and joint approaches that utilize the BIO and +/-/0 tagging schemes, while "collapsed" is the model following the collapsed tagging scheme (Figure 2(a)).

UNIFIED (Li et al., 2019) is the current state-of-the-art model on targeted sentiment analysis[3]. It contains two stacked recurrent neural networks enhanced with multi-task learning and adopts the collapsed tagging scheme.

We also compare our multi-target extractor with the following method:

DE-CNN (Xu et al., 2018) is the current state-of-the-art model on target extraction, which combines a double embeddings mechanism with convolutional neural networks (CNNs)[4].

Finally, the polarity classifier is compared with the following methods:

MGAN (Fan et al., 2018) uses a multi-grained attention mechanism to capture interactions between targets and sentences for polarity classification.

TNet (Li et al., 2018) is the current state-of-the-art model on polarity classification, which consists of a multi-layer context-preserving network architecture and uses CNNs as feature extractor[5].

4.3 Main Results

We compare models under either the sequence tagging scheme or the span-based labeling scheme, and show the results in Table 2. We denote our approach as "SPAN", and use $\text{BERT}_{\text{LARGE}}$ as backbone networks for both the "TAG" and "SPAN" models to make the comparison fair.

Two main observations can be obtained from the Table. First, despite that the "TAG" baselines already outperform previous best approach ("UNIFIED"), they are all beaten by the "SPAN" methods. The best span-based method achieves 1.55%, 0.94% and 3.43% absolute gains on three datasets compared to the best tagging method, indicating the efficacy of our extract-then-classify framework. Second, among the span-based methods, the SPAN-pipeline achieves the best performance, which is similar to the results of Mitchell et al. (2013); Zhang et al. (2015). This suggests that there is only a weak connection between target extraction and polarity classification. The conclusion is also supported by the result of SPAN-collapsed method, which severely drops across all datasets, implying that merging polarity labels into target spans does not address the task effectively.

[3]https://github.com/lixin4ever/E2E-TBSA

[4]https://www.cs.uic.edu/~hxu/

[5]https:// github.com/lixin4ever/TNet

Model	LAPTOP	REST	TWITTER
DE-CNN	81.59	-	-
TAG	**85.20**	**84.48**	73.47
SPAN	83.35	82.38	**75.28**

Table 3: F1 comparison of different approaches for target extraction.

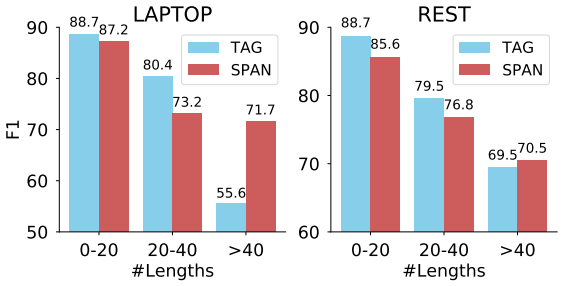


Figure 5: F1 on LAPTOP and REST w.r.t different sentence lengths for target extraction.

4.4 Analysis on Target Extraction

To analyze the performance on target extraction, we run both the tagging baseline and the multi-target extractor on three datasets, as shown in Table 3. We find that the BIO tagger outperforms our extractor on LAPTOP and REST. A likely reason for this observation is that the lengths of input sentences on these datasets are usually small (e.g., 98% of sentences are less than 40 words in REST), which limits the tagger's search space (the power set of all sentence words). As a result, the computational complexity has been largely reduced, which is beneficial for the tagging method.

In order to confirm the above hypothesis, we plot the F1 score with respect to different sentence lengths in Figure 5. We observe that the performance of BIO tagger dramatically decreases as the sentence length increases, while our extractor is more robust for long sentences. Our extractor manages to surpass the tagger by 16.1 F1 and 1.0 F1 when the length exceeds 40 on LAPTOP and REST, respectively. The above result demonstrates that our extractor is more suitable for long sentences due to the fact that its search space only increases linearly with the sentence length.

Since a trade-off between precision and recall can be adjusted according to the threshold γ in our extractor, we further plot the precision-recall curves under different ablations to show the effects of heuristic multi-span decoding algorithm. As can be seen from Figure 6, ablating the length

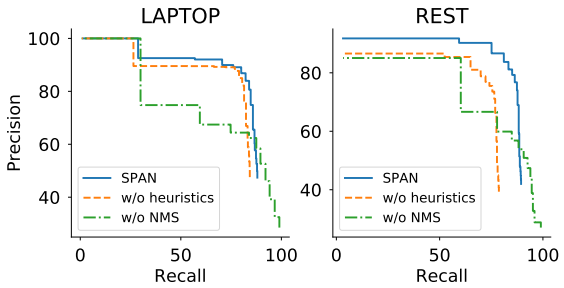


Figure 6: Precision-recall curves on LAPTOP and REST for target extraction. "NMS" and "heuristics" denote the non-maximum suppression and the length heuristics in Algorithm 1.

heuristics results in consistent performance drops across two datasets. By sampling incorrect predictions we find that there are many targets closely aligned with each other, such as "*perfect* ***[size]***$_+$ *and* ***[speed]***$_+$", "***[portions]***$_+$ *all at a reasonable* ***[price]***$_+$", and so on. The model without length heuristics is very likely to output the whole phrase as a single target, thus being totally wrong. Moreover, removing the non-maximum suppression (NMS) leads to significant performance degradations, suggesting that it is crucial to prune redundant spans that refer to the same text.

4.5 Analysis on Polarity Classification

To assess the polarity classification subtask, we compare the performance of our span-level polarity classifier with the CRF-based tagger in Table 5. The results show that our approach significantly outperforms the tagging baseline by achieving 9.97%, 8.15% and 15.4% absolute gains on three datasets, and firmly surpasses previous state-of-the-art models on LAPTOP. The large improvement over the tagging baseline suggests that detecting sentiment with the entire span representation is much more beneficial than predicting polarities over each word, as the semantics of the given target has been fully considered.

To gain more insights on performance improvements, we plot the accuracy of both methods with respect to different target lengths in Figure 7. We find that the accuracy of span-level classifier only drops a little as the number of words increases on the LAPTOP and REST datasets. The performance of tagging baseline, however, significantly decreases as the target becomes longer. It demonstrates that the tagging method indeed suffers from the sentiment inconsistency problem when it comes to multi-word target entities. Our

Sentence	TAG	SPAN
1. I thought the transition would be difficult at best and would take some time to fully familiarize myself with the new **[Mac ecosystem]$_0$**.	[ecosystem]$_+$ (✗)	[Mac ecosystem]$_0$
2. I would normally not finish the **[brocolli]$_+$** when I order these kinds of food but for the first time, every piece was as eventful as the first one... the **[scallops]$_+$** and **[prawns]$_+$** was so fresh and nicely cooked.	[brocolli]$_-$ (✗), [scallops and prawns]$_+$ (✗), [food]$_0$ (✗)	[brocolli]$_+$, [scallops]$_+$, [prawns]$_+$
3. I like the **[brightness]$_+$** and **[adjustments]$_+$**.	[brightness]$_+$, [adjustments]$_+$	[brightness]$_+$, None (✗)
4. The **[waiter]$_-$** was a bit unfriendly and the **[feel]$_-$** of the restaurant was crowded.	[waiter]$_-$, [feel]$_-$	[waiter]$_-$, None (✗)
5. However, it did not have any scratches, zero **[battery cycle count]$_+$** (pretty surprised), and all the **[hardware]$_+$** seemed to be working perfectly.	[battery cycle count]$_0$ (✗), [hardware]$_+$	[battery cycle count]$_+$, [hardware]$_+$
6. I agree that dining at **[Casa La Femme]$_-$** is like no other dining experience!	[Casa La Femme]$_+$ (✗)	[Casa La Femme]$_-$

Table 4: Case study. The extracted targets are wrapped in brackets with the predicted polarities given as subscripts. Incorrect predictions are marked with ✗.

Model	LAPTOP	REST	TWITTER
MGAN	75.39	-	-
TNet	76.54	-	-
TAG	71.42	81.80	59.76
SPAN	**81.39**	**89.95**	**75.16**

Table 5: Accuracy comparison of different approaches for polarity classification.

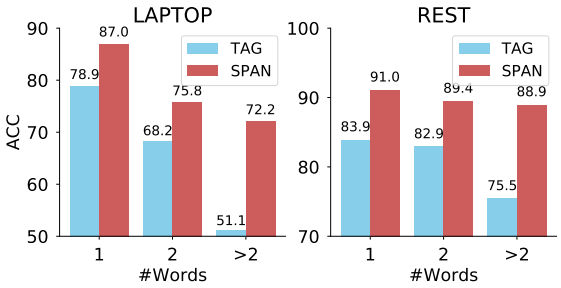


Figure 7: Accuracy on LAPTOP and REST w.r.t different number of target words for polarity classification.

span-based method, on the contrary, can naturally alleviate such problem because the polarity is classified by taking all target words into account.

4.6 Case Study

Table 4 shows some qualitative cases sampled from the pipeline methods. As observed in the first two examples, the "TAG" model incorrectly predicts the target span by either missing the word "*Mac*" or proposing a phrase across two targets ("*scallps and prawns*"). A likely reason of its failure is that the input sentences are relatively longer, and the tagging method is less effective when dealing with them. But when it comes to shorter inputs (e.g., the third and the fourth examples), the tagging baseline usually performs better than our approach. We find that our approach may sometimes fail to propose target entities (e.g., "*adjustments*" in (3) and "*feel*" in (4)), which is due to the fact that a relatively large γ has been set. As a result, the model only makes cautious but confident predictions. In contrast, the tagging method does not rely on a threshold and is observed to have a higher recall. For example, it additionally predicts the entity "*food*" as a target in the second example. Moreover, we find that the tagging method sometimes fails to predict the correct sentiment class, especially when the target consists of multiple words (e.g., "*battery cycle count*" in (5) and "*Casa La Femme*" in (6)), indicating the tagger can not effectively maintain sentiment consistency across words. Our polarity classifier, however, can avoid such problem by using the target span representation to predict the sentiment.

5 Conclusion

We re-examine the drawbacks of sequence tagging methods in open-domain targeted sentiment analysis, and propose an extract-then-classify framework with the span-based labeling scheme instead. The framework contains a pre-trained Transformer encoder as the backbone network. On top of it, we design a multi-target extractor for proposing multiple candidate targets with an heuristic multi-span decoding algorithm, and introduce a polarity classifier that predicts the sentiment towards each candidate using its summarized span representation. Our approach firmly outperforms the sequence tagging baseline as well as previous state-of-the-art methods on three benchmark datasets. Model analysis reveals that the main performance improvement comes from the span-level polarity classifier, and the multi-target extractor is more

suitable for long sentences. Moreover, we find that the pipeline model consistently surpasses both the joint model and the collapsed model.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback. We also thank Li Dong for his helpful comments and suggestions. This work was supported by the National Key Research and Development Program of China (2016YFB1000101).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of EMNLP*.

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*.

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 ACL*.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of EMNLP*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. *arXiv preprint arXiv:1805.04787*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of IJCAI*.

Binxuan Huang and Kathleen Carley. 2018. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of EMNLP*.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of EMNLP*.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. In *Proceedings of ACL*.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019. A unified model for opinion target extraction and target sentiment prediction. In *Proceedings of AAAI*.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of EMNLP*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of EMNLP*.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of EMNLP*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of SemEval-2016*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of SemEval 2015*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of SemEval-2014*.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

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

Azriel Rosenfeld and Mark Thurston. 1971. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, (5):562–569.

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

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning crf for supervised aspect extraction. In *Proceedings of the ACL*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING*.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

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 *Proceedings of NIPS*.

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

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016a. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of EMNLP*.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016b. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of EMNLP*.

Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Double embeddings and cnn-based sequence labeling for aspect extraction. In *Proceedings of ACL*.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of EMNLP*.

A Complexity Analysis, Design Choices, and Implementation Notes

Relation to Main Text. This appendix provides detailed derivations and algorithmic explanations for the span-based extractor (§3.2) and polarity classifier (§3.3) in the main paper, together with complexity analysis, theoretical properties, and implementation notes.

A.1 Task Setup and Notation

Let the sentence length be N. The BIO-style label set has size $L = 5$ (B+, B−, I+, I−, O). A span-based view predicts two position-wise vectors:

$$S \in \mathbb{R}^N \quad \text{(start logits/probabilities)}, \tag{1}$$

$$E \in \mathbb{R}^N \quad \text{(end logits/probabilities)}. \tag{2}$$

We select $\mathrm{TopM}(S)$ and $\mathrm{TopM}(E)$, pair valid (s, e) with $e \geq s$, remove conflicts, and keep at most K spans. Unless otherwise stated, M, K are small constants independent of N.

BIO-to-span correspondence. Any BIO-valid labeling $y_{1:N} \in L^N$ induces a set of non-overlapping, polarity-annotated spans $\{(s_m, e_m, \pi_m)\}_{m=1}^{M^\star}$ with $\pi_m \in \{+, -\}$; conversely, any such span set induces at least one BIO sequence. This bijection underlies our span-level decoding.

A.2 From Traditional CRF to Our Method

CRF (token labeling). Let $\phi_t(\ell)$ be unary scores and $T_{\ell',\ell}$ transitions. The sequence score is

$$s(x, y) = \sum_{t=1}^{N} \phi_t(y_t) \; + \; \sum_{t=2}^{N} T_{y_{t-1}, y_t}. \tag{3}$$

Forward recursion and Viterbi decoding both run in

$$\text{time} = \mathcal{O}(N\,|L|^2) \tag{4}$$

$$= \mathcal{O}(N) \quad (\text{for fixed } |L|{=}5). \tag{5}$$

The label-space size 5^N is combinatorial and does not determine the actual decoding cost in CRFs.

Naïve span extraction. Let $\mathcal{C} = \{(i, j) : 1 \leq i \leq j \leq N\}$ with $|\mathcal{C}| = \Theta(N^2)$. Methods that score every $(i, j) \in \mathcal{C}$ are $\mathcal{O}(N^2)$.

Our boundary-first span decoding. We restrict search to boundary sets $S_M = \mathrm{TopM}(S)$ and $E_M = \mathrm{TopM}(E)$. Pair only feasible (s, e) and select a conflict-free subset. The decoding stage is $\mathcal{O}(N)$ when M, K are constants. This linear-time property corresponds to the reduced *extractive* search space stated in the main text, in contrast to the combinatorial tagging space.

A.3 Detailed Complexity (Time and Space)

Let $\mathcal{T}(N; M, K)$ denote time and $\mathcal{S}(N; M, K)$ memory:

$$\mathcal{T}(N; M, K) = \underbrace{\mathcal{O}(N)}_{\text{compute } S,E \text{ and } \mathrm{TopM}} + \underbrace{\mathcal{O}(M^2)}_{\text{pairing}} + \underbrace{\mathcal{O}(M^2)}_{\text{conflict resolution}} = \mathcal{O}(N), \tag{6}$$

$$\mathcal{S}(N; M, K) = \underbrace{\mathcal{O}(N)}_{\text{store } S,E} + \underbrace{\mathcal{O}(M^2)}_{\text{candidates}} + \underbrace{\mathcal{O}(K)}_{\text{output}} = \mathcal{O}(N). \tag{7}$$

If $M = \Theta(N^\alpha)$ for some $\alpha > 0$, then

$$\mathcal{T}(N; M, K) = \mathcal{O}\big(N + N^{2\alpha}\big). \tag{8}$$

Keeping M constant is therefore essential for linear-time decoding.

A.4 Pairing Score, Validity, and Conflict Resolution

For a candidate span (s, e),

$$u(s, e) = a_s\, S[s] + a_e\, E[e] - \lambda\,(e - s + 1) + r(s, e), \tag{9}$$

$$\mathcal{R} = \big\{\, (s, e) \in S_M \times E_M \;\big|\; e \geq s,\; S[s] + E[e] \geq \gamma \,\big\}. \tag{10}$$

Here $a_s, a_e \geq 0$ weight boundary confidence, $\lambda \geq 0$ regularizes length, and $r(s, e)$ is a bounded span-level term (e.g., local context) that does not scale with N. The length regularizer mirrors the heuristic term in Algorithm 1 of the main paper and prevents long multi-entity phrases (e.g., "food but the service") from collapsing into a single target.

Overlap models. View spans as intervals on a line. Conflicts are edges in an interval graph. Two common consistency rules: (i) *non-overlap* (independent set in the interval graph); (ii) *soft overlap* via a penalty or suppression.

NMS vs. optimal interval selection. Greedy NMS is linear in the candidate count and effective in practice. If exact *non-overlapping* selection is required, weighted interval scheduling yields the optimum via the DP

$$\mathrm{OPT}(j) = \max\big\{\mathrm{OPT}(j-1), \\ u(j) + \mathrm{OPT}\big(p(j)\big)\big\}, \quad (11)$$

on spans sorted by end time, where $p(j)$ is the last index compatible with j. This costs $\mathcal{O}(M \log M) + \mathcal{O}(M)$ (sorting + DP); with constant M, the overall decoding remains $\mathcal{O}(N)$.

Token-level overlap metrics. For spans $A = [s, e]$, $B = [s', e']$:

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (12)$$

$$\mathrm{F1}(A, B) = \frac{2\,|A \cap B|}{|A| + |B|}. \quad (13)$$

Either can drive suppression or serve as a compatibility predicate.

A.5 Theoretical Properties

Span/BIO consistency. Let $\mathcal{Y}_{\mathrm{BIO}}$ be the set of BIO-valid sequences and $\mathcal{I}$ the family of non-overlapping, polarity-labeled interval sets. The mapping $\psi : \mathcal{Y}_{\mathrm{BIO}} \to \mathcal{I}$ (collapse B/I runs) is surjective; restricting to canonical encodings makes it bijective. Thus reasoning at span level preserves the feasible set of outputs.

Coverage under Top-M. Proposition 1. If every gold span $(s^\star, e^\star)$ satisfies

$$s^\star \in S_M, \;\; e^\star \in E_M, \quad S[s^\star] + E[e^\star] \geq \gamma, \quad (14)$$

then $(s^\star, e^\star) \in \mathcal{R}$. Its survival depends only on the conflict rule and the ordering by u.

Sketch. Feasibility is by construction. Selection is monotone in u for NMS and solved exactly by the interval DP for non-overlap. □

Margin-based sufficiency. Let Δ_s be the smallest start-score margin between a gold start and any non-gold position; define Δ_e analogously. If M exceeds the number of impostor positions within margin Δ_s (resp. Δ_e), all gold endpoints fall in S_M, E_M. Any fixed positive margins imply existence of a finite M that guarantees coverage.

Error decomposition. End-to-end span errors decompose into (i) boundary miss-inclusion ($s^\star \notin S_M$ or $e^\star \notin E_M$); (ii) feasibility violations ($S[s] + E[e] < \gamma$); (iii) conflict pruning. This aligns diagnostics with controllable knobs M, γ and the conflict policy.

Relationship to CRF objectives. CRFs maximize a globally normalized token-transition potential, which implicitly encodes span validity via transitions. Our decoder optimizes an object-level potential $u(s, e)$ over intervals, making span length, boundary confidence, and overlap control explicit. When $r(s, e) \equiv 0$ and a_s, a_e arise from calibrated token marginals, the span score approximates a product-of-marginals decision rule at boundaries.

A.6 Top-M Selection and Implementation Notes

TopM can be implemented by selection in expected $\mathcal{O}(N)$ time (e.g., Quickselect) or by a heap in $\mathcal{O}(N + M \log M)$. On accelerators, segmented reductions compute TopM per sequence in parallel. Deterministic tie-breaking (e.g., smaller indices first) improves reproducibility. To control memory, compute S, E in fp16 with loss-scaling and keep indices in 32-bit ints.

A.7 Pseudocode

Algorithm 2 Top-M/K Constrained Span Decoding (Linear in N)

Require: start scores $g_s[1{:}N]$, end scores $g_e[1{:}N]$, threshold γ, integers M, K

function TopMIndices(**g**)
  return indices of the M largest elements in **g** ▷ $\mathcal{O}(N)$ expected
$S \leftarrow$ TopMIndices(g_s); $E \leftarrow$ TopMIndices(g_e) ▷ $\mathcal{O}(N)$
$R \leftarrow \varnothing$ ▷ candidate spans
for all $s \in S$ **do**
  for all $e \in E$ **do**
    if $e \geq s$ **and** $g_s[s] + g_e[e] \geq \gamma$ **then**
      $u \leftarrow a_s g_s[s] + a_e g_e[e] - \lambda(e - s + 1) + r(s, e)$
      $R \leftarrow R \cup \{(s, e, u)\}$
$O \leftarrow \mathrm{NMS}\_and\_KeepTopK(R, K)$ ▷ or interval-DP for exact non-overlap
return O

A.8 Training Objective, Calibration, and Stability

We jointly supervise boundaries and per-span decisions:

$$\mathcal{L} = \mathrm{CE}\big(y^{\mathrm{start}}, S\big) + \mathrm{CE}\big(y^{\mathrm{end}}, E\big) + \lambda_{\mathrm{span}}\, \mathcal{L}_{\mathrm{span}}. \tag{15}$$

Boundary calibration (e.g., temperature scaling) aligns S, E so that γ has stable semantics across domains and lengths. For numerical stability, compute attention and softmax in log-space for long sequences; normalize span representations v (e.g., LayerNorm) before classification. Optional class weights in $\mathcal{L}_{\mathrm{span}}$ mitigate polarity imbalance.

A.9 Hyperparameters and Reproducibility

Hyperparameter Settings. We set $M{=}20$, $K{=}10$, and tune γ on the validation set for each dataset. These constants guarantee linear-time decoding with a small, high-confidence candidate pool.

Training Setup. We use $\mathrm{BERT}_{\mathrm{LARGE}}$ (24 layers, hidden size 1024) as the backbone. Optimization uses Adam with learning rate $2{\times}10^{-5}$, linear warmup over the first 10% steps, batch size 32, dropout 0.1, and 3 epochs. Experiments run on a single NVIDIA P100 GPU. Code follows the public implementation.[6]

A.10 Edge Cases and Robustness

- **Subword constraints.** Enforce starts at first subword and ends at last subword to preserve span/BIO consistency under tokenization.
- **Laminar regimes.** When conflicts are laminar (pure nesting, no crossing), greedy suppression that keeps the heaviest of a nest is optimal; otherwise use interval DP if exactness is required (still $\mathcal{O}(N)$ overall for constant M).
- **Length priors.** Choosing $\lambda > 0$ induces an exponential penalty on long spans in a maximum-score sense, acting as a simple regularizer against boundary noise.

[6] https://github.com/huminghao16/SpanABSA

A.11 Summary and Conclusions

CRF offers globally normalized token transitions with linear-time decoding for fixed L, while our span-based decoder operates directly on objects of interest and retains linear-time complexity by restricting to high-confidence boundaries and solving interval conflicts on a constant-size candidate set.