The University of Arizona at SemEval-2021 Task 10: Applying Self-training, Active Learning and Data Augmentation to Source-free Domain Adaptation

Xin Su
School of Information
University of Arizona
xinsu@email.arizona.edu

Yiyun Zhao
Department of Linguistics
University of Arizona
yiyunzhao@email.arizona.edu

Steven Bethard
School of Information
University of Arizona
bethard@arizona.edu

Abstract

This paper describes our systems for negation detection and time expression recognition in SemEval 2021 Task 10, Source-Free Domain Adaptation for Semantic Processing. We show that self-training, active learning and data augmentation techniques can improve the generalization ability of the model on the unlabeled target domain data without accessing source domain data. We also perform detailed ablation studies and error analyses for our time expression recognition systems to identify the source of the performance improvement and give constructive feedback on the temporal normalization annotation guidelines.

1 Introduction

Unsupervised Domain Adaptation (UDA) is a task that generalizes knowledge acquired from a model trained on labeled data in one domain (source domain) to unlabeled data in a different domain (target domain). Conventional UDA algorithms usually require access to both source-domain and target-domain data (Ganin et al., 2016; Glorot et al., 2011; Chen et al., 2012; Louizos et al., 2016). However, sharing source-domain data is often not practical for clinical texts due to their highly sensitive personal information and complex data use agreement procedures (Laparra et al., 2020). To overcome this difficulty, Laparra et al. (2020) propose a new task of source free domain adaptation (SFDA) where only models trained on source-domain data are shared, which allows the possibility of using the information from the source-domain while reducing private information leakage. The biggest challenge of this task is to transfer task-related information embedded in the trained models.

Our team participated in both subtasks of SemEval 2021 Task 10 (Laparra et al., 2021), Source Free Domain Adaptation for Semantic Processing: negation detection and time expression recognition. For both tasks, participants were given a RoBERTa model (Liu et al., 2019) fine-tuned on the source-domain, and asked to make predictions in the target-domain.

The goal of the negation detection task is to predict whether an event in a sentence is negated by its context. This is a binary sentence classification task. For example, given the event *diarrhea* and the sentence *Has no diarrhea and no new lumps or masses*, the goal is to predict that *diarrhea* is negated by its context.

The goal of time expression recognition sub-task (Laparra et al., 2018) is to recognize time expressions in the target domain. This is a named entity recognition (NER) task. The number of entity types (inside–outside–beginning format) is 65. Entity types in this task are formally defined time entity types from the Semantically Compositional Annotation of Time Expressions (SCATE) (Bethard and Parker, 2016) annotation schema. For example, in *2021-02-19*, *2021* will be labeled as Year, *02* will be labeled as Month-Of-Year and *19* will be labeled as Day-Of-Month.

We investigate self-training, active learning, and data augmentation techniques on negation detection and time expression recognition under the SFDA setting. Our contributions are:

1. We demonstrate that simple self-training over a small portion of the target domain data can effectively improve the performance of the negation detection model.
2. We demonstrate that active learning with data augmentation can significantly improve time expression recognition performance when selected examples are accurately annotated.
3. We perform ablation studies for the time expression recognition systems to analyze where the performance improvement comes from.
4. We analyze our annotation errors for the time recognition task and give constructive feed-

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 458–466
Bangkok, Thailand (online), August 5–6, 2021.

back on the annotation guideline and schema.

2 System Description

The source-domain models for both subtasks are RoBERTa-base models with linear classification output layers, implemented via the Huggingface Transformers library (Wolf et al., 2020), using RobertaForSequenceClassification for negation, and RobertaForTokenClassification for time.

The input to the models is a sequence tokenized by Byte-Pair Encoding (BPE). Following the conventions of the RoBERTa model input format, two special tokens <s> and </s> are inserted at the beginning and end of the sequence, respectively. In the negation detection task, targeted events are denoted with two special tokens <e> and </e> that are inserted before and after the event. For example, the sentence *Has no diarrhea and no new lumps or masses* with event *diarrhea* will be converted to <s>*Has no* <e>*diarrhea*</e> *and no new lumps or masses.*</s>. The model output for negation detection is whether the target event is negated and the model output for time expression detection is the labels for each input tokens.

2.1 Negation Detection System

We employ a simple self-training (Yarowsky, 1995) approach that fine-tunes the model with its own predictions on the unlabeled dataset. We start with the pre-trained source-domain model, M. Then, for each self-training iteration:

1. We initialize an empty training set, L.
2. We use M to label the target domain data.
3. If an instance is labeled with a probability above a threshold τ, we add it to L with the predicted label as its pseudo label.
4. We fine-tune M on L.

When the source-domain model predictions are the same for two consecutive iterations or the number of iterations of self-training is greater than the predefined maximum number, self-training stops. Note that the training set L is reinitialized at each iteration, and the model is iteratively fine-tuned.

2.2 Time Expression Detection System

Our approach combines active-learning (Cohn et al., 1996) and data-augmentation (Simard et al., 2003). We start with the pre-trained source-domain model, M_0, a copy of the pre-trained source-domain model, M, and initialize an empty training set L. Then, for each iteration:

1. We select the k instances where M is most uncertain, manually annotate them, and add them to L. (Details in section 2.2.1.)
2. We augment each manually annotated instance with n new examples and add them to L. (Details in section 2.2.2.)
3. We re-initialize M to M_0 and fine-tune on L.

We repeat this process i times. Note that the training set L is built cumulatively, and M is reinitialized on each iteration.

2.2.1 Active Learning

We use active learning methods to manually label the most uncertain examples of the model in the target domain. We believe that it is not practical to manually label the entire target domain dataset during the test phase. This requires sufficient expertise and time from annotators (we show later that it is very difficult to understand annotation guidelines in a short time). Otherwise, low-quality annotations will hurt the performance of the model. In each iteration, we select the top k target domain sentences with the highest uncertainty scores to manually annotate. We define the uncertainty score of an example as the sum of the model's prediction's entropy for each token within the sentence. Manual annotation follows the SCATE annotation guidelines released by the organizers.

The annotators were the first two authors of this paper, a Linguistic PhD student and a Information PhD student. During the annotation process, we first individually annotated examples and then resolved annotation differences through discussion. Our first exposure to the SCATE annotation schema was approximately 10 days before the start of the test phase, when we began reading the guidelines and posting questions on the Google forum. We used gold annotations from the development set (on the news domain) to simulate the annotation process during the practice phase. We believe this is similar to most real-world SFDA situations, where the person applying the model on the target domain is unfamiliar with the annotation guidelines and has limited time to learn them.

2.2.2 Data Augmentation

Inspired by Miao et al. (2020), we applied data augmentation to increase the size of our training set beyond what can be achieved by manual annotation, and to improve the generalization of the model. For each time entity that we manually annotated, we automatically generated new training examples by

Task	Type	Domain	#	# of labeled	Open to participants
Negation	Train	-	-	all	✗
Negation	Dev	Clinical	8431 sentences	5545 sentences	✓
Negation	Test	Clinical	622703 sentences	9580 sentences	✓
Time	Train	-	-	all	✗
Time	Dev	news	99 documents	all	✓
Time	Test	food security	48 documents	17 documents	✓

Table 1: Data summary for negation detection and time expression recognition tasks

substituting original time entities with entities of the same type randomly sampled from a predefined entity candidates pool. We generate up to n new sequences (the size of the pool may be less than n for some entities). For example, if we manually annotate the three time entities from *2021-02-19* (*2021*: Year, *02*: Month-Of-Year, *19*: Day-Of-Month) in a sentence, after data augmentation, it can generate up to $n \times 3$ additional sequences with different years, months and days (e.g., *2020-10-05*). The entity candidates pool is created based on the time entities in the development set and the annotation guideline. We filtered out entities that do not appear in the unlabeled test set data during the testing phase. See appendix A.1 for the final pool.

3 Data

All data used is in English. Both subtasks had training, development and test data, each representing different domains. As participants, we did not have access to the training set. The training sets are used by organizers to fine-tune the pretrained RoBERTa-base models to obtain the source-domain models. We used the source-domain models and development sets to develop source-free domain adaptation systems during the practice phase, and tested our systems during test phase. We summarize the data in table 1.

4 Experiments

The organizers provided two baseline models for each task: the source-domain model, and the source-domain model fine-tuned on the development set. The official evaluation metric is the F1 score. Precision and recall scores are also reported.

4.1 Negation Detection

In the testing phase, we first fine-tuned the source-domain model on the labeled development set. Although the domains of the development set and the test set are different, they are both clinically relevant data, so we believed that fine-tuning the model on the development set could improve its performance on the test set. Because of time and hardware constraints, we randomly sampled 3000 instances from the 622,703 test set instances as unlabeled data for self-training. We used the same hyperparameters for fine-tuning the source-domain model on the development set and self-training the fine-tuned model on the randomly sampled test data. All the hyperparameters are shown in table 4 in appendix A.2. Our submission ranked 2nd. Table 2 shows that our system outperformed both baseline models provided by the organizers.

Strategy	F1	Precision	Recall
Test Phase			
SD	0.660	0.917	0.516
SD + FT	0.730	0.908	0.611
SD + FT + ST	**0.767**	0.880	0.680

Table 2: The performance of the negation detection systems during test phase. SD is the source-domain model. FT is fine-tuning on the development set. ST is self-training on test set.

4.2 Time Expression Recognition

We did not fine-tune the source-domain model on the development set during the test phase. The development set is from the newswire domain, while the test set is from the food security domain. We though that there might be a large difference between these two domains. Fine-tuning on a different domain may hurt the performance of the model on the test set. As with the code provided by the organizer, we used the sentencizer from Spacy (Honnibal et al., 2020) to split the input documents into sentences and used them as inputs to the model. All the hyperparameters are shown in table 5 in

#	Strategy	F	P	R
Test Phase				
1	SD (baseline)	0.794	0.849	0.746
2	SD + FT (baseline)	**0.804**	0.827	0.782
3	SD + AL (32*5) + DA (5) + Manual Annotations	0.795	0.783	0.807
Post-Evaluation				
4	SD + AL (32*5) + DA (5) + Manual Annotations (fixed seasonal(ly))	0.837	0.824	0.850
5	SD + AL (32*5) + DA (5) + Gold Annotations	0.955	0.945	0.965
6	SD + FT + AL (32*5) + DA (5) + Gold Annotations	**0.959**	0.949	0.969
7	SD + AL (32*5) + Gold Annotations	0.890	0.893	0.887
8	SD + AL (16*5) + DA + Gold Annotations	0.929	0.918	0.941
9	SD + AL (8*5) + DA + Gold Annotations	0.900	0.880	0.920
10	SD + AL (4*5) + DA + Gold Annotations	0.877	0.860	0.894
11	SD + AL (4*5) + Gold Annotations	0.851	0.846	0.855

Table 3: The performance of the time expression recognition systems during the test and post-evaluation phases. SD is the source-domain model. FT is fine-tuning on the development set. AL (k*i) is active learning with k samples and i iterations. DA (n) is data augmentation with n generated examples.

appendix A.2. Our submitted system ranked 6th. Table 3 shows that our submitted system's performance (row 3) is no better than the best baseline model (row 2) provided by the organizers.

To investigate the reasons for the lower-than-expected test performance, we used the gold annotations in the test set for our post-evaluation runs (row 5-11 in table 3). Note that performance for these rows will be artificially inflated, since up to 160 of the 926 test sentences were included in the system's training data. Nonetheless, we see that by using the gold annotations instead of our manual annotations (row 5 vs row 3 in table 3), the performance of our system improved by 0.160 F1 score. This seems to suggest that our system can improve its generalization ability if we can accurately label the target domain data.

We further analyze where the performance improvement comes from in section 5 and provide a detailed analysis of our annotation errors and give feedback on annotation guidelines in section 6.

5 Time Expressions Ablation Study

Effect of Fine-Tuning on Dev Data From the baseline models' performances (row 1 vs row 2 in table 3), we can see that the test performance of the model fine-tuned on the development set is slightly better than the pure source-domain model (+.010 F1 score). To verify if this is also true for our active learning system, we add the fine-tuning strategy to our system (row 6 in table 3) and run the system on the labeled portion of the test set. The results (row 5 vs row 6 in 3) indicate that fine-tuning on the additional domain continues to help a bit (+.004 F1 score) even when followed by active learning.

Effect of Data Augmentation We also investigate the contribution of our data augmentation strategy, removing it from our system and running on the labeled test set. The result shows that data augmentation brings a +.065 F1 score improvement to our system (row 7 vs row 5 in table 3). This indicates that data augmentation was a major source of performance improvements.

Effect of Size of Annotation Data In real-world use cases, we often want to keep the size of annotated data as small as possible, since annotation is time consuming and error-prone. To understand how our system performs with less manual annotated examples, we reduce the number of sentences to be annotated at each active learning iteration to 16, 8 and 4 resulting in rows 8, 9 and 10 in table 3. The results show that with only 20 correctly annotated sentences but incorporating data augmentation (row 10), our system outperform the best baseline model (row 2) by .073 F1. If we remove data augmentation from this model (row 11) its performance declines, but still outperforms the best baseline model by .047 F1.

6 Time Expressions Annotation Analysis

Though gold annotations led to large performance improvements, the annotation for this task is challenging for untrained people. Through reading the SCATE annotation guidelines and posting questions on the share task google group, our team annotated 160 sentences of which 48 sentences were in the labeled portion of the test set. We annotated 13008 tokens in total (including padding tokens) and our overall accuracy on the gold 48 sentences is 0.991 for all categories and 0.785 excluding the category O. We report detailed performance for each of the entity types in table 6 in appendix A.3.

We found several annotation patterns where our team consistently disagreed with the gold annotations. Our errors can be broadly attributed to two reasons: misinterpretation/underspecification of annotation guidelines, and ambiguity of the phrases.

Errors from misinterpretations/underspecification of annotation schema : We annotated the token *seasonal(ly)* (e.g., *seasonal progress*, *seasonal rainfall*) as Calendar-Interval instead of Season-Of-Year as we thought Season-Of-Year is applied to seasons that are explicitly specified (such as summer). We considered *seasonal* similar to *weekly*, both referring to an interval unspecified. However, Season-Of-Year could be applied to very broad categories such as dry seasons and rainfall seasons including seasons that are not specified. Also, *seasonal* unlike *weekly/monthly/yearly* only refers to one season of a year instead of every season of a year. Due to the ubiquity of this token in the dataset, this error affects our overall performance. Correcting the annotation of this particular token leads to +.042 F1 score improvement (row 4 vs row 3 in table 3). Another erroneous pattern is that we double-annotated the phrase such as *from ... to ...* and *between ... to ....* Specifically, we annotated both adpositions instead of choosing the first adposition only. Finally, we also annotated more modifiers than the gold annotations. For instance, we annotated *marketing* in *marketing year* and *long* in *long dry Jiaal Season* as 'Modifier' instead of the category 'O'. It turns out that the category of modifier in the gold annotation is a closed category, and only a specific set of tokens are considered modifiers.

Errors from ambiguity within phrases Some phrases allow multiple interpretations that lead to different ways of annotations. For instance, confusion between 'Period' and 'Calendar Interval' occurred frequently (e.g., we annotated *weeks* in *recent weeks* as 'Period' rather than 'Calendar-Interval'). Although *"/"* between seasons is commonly annotated as 'I-Season-Of-Year' in the gold annotations, we found different roles it might play in specific contexts. For example, if *"/"* is used between two terms that refer to the same season, then it should be annotated as 'I-Season-Of-Year'; if it is used between two non-adjacent seasons, it should be annotated as 'Union'; and if it is used between two adjacent seasons, then it could be annotated as 'Between' or 'Union'. Thus, the correct annotation requires a surprising amount of external knowledge about Ethiopian season terms. In fact, there are still cases that remained uncertain: For example, *Xaran* refers to seasonal rains from April through September and *Xagaa* refers to the second dry season (July to September) and when the two tokens joined by *"/"* it is difficult to interpret the meaning of *"/"*. We also found the conjunction *and* causes ambiguity. For example, *and* in *rains in May and August* could be considered as an operator over months (i.e., rains in Union(May, August)) or an operator over rains (i.e., Union(rains in May, rains in August)). The former understanding requires annotating *and* whereas the latter does not despite the fact that the two interpretations are essentially semantically equivalent. Lastly, we also found the particle *the* is difficult to annotate. For instance, *the* in *the month* depending on the context may be annotated as *this* or *last* and sometimes the context may not be clear enough to tell the differences.

Our annotation analysis leads to several suggestions for the annotation schema and the documentation. Our errors in the first category indicate some potential helpful updates can be made such as including more examples in categories (e.g., 'Season-Of-Year'), explicit documentation of whether the certain category is closed or open as well as the specific manner to deal with multi-word phrases or even circumpositions. The second category of errors, however, might involve the refinement of the annotation schema. For example, maybe 'Between' and 'Union' can be unified together, and 'Period' can be merged into 'Calendar Interval' or confined to an explicit set of circumstances.

7 Conclusion

Our overall rank (by F1 score) for negation detection task was 2nd and for time expression recogni-

tion was 6th.

Our results suggest that simple self-training can be used in sentence-level SFDA tasks to improve a trained model's performance on a new domain. As for token-level tasks, our analysis shows that both active learning and data augmentation can bring significant performance improvements, but the premise is that the data in active learning can be correctly annotated. Our analysis and feedback could also be used to improve the SCATE annotation guidelines/schema in future work.

Acknowledgments

Research reported in this publication was supported by the National Library of Medicine of the National Institutes of Health under Award Numbers R01LM012918 and R01LM010090. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Steven Bethard and Jonathan Parker. 2016. A semantically compositional annotation scheme for time normalization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3779–3786, Portorož, Slovenia. European Language Resources Association (ELRA).

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1627–1634, Madison, WI, USA. Omnipress.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 513–520, Madison, WI, USA. Omnipress.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Egoitz Laparra, Steven Bethard, and Timothy A Miller. 2020. Rethinking domain adaptation for machine learning over clinical language. *JAMIA Open*, 3(2):146–150.

Egoitz Laparra, Xin Su, Yiyun Zhao, Ozlem Uzuner, Timothy Miller, and Steven Bethard. 2021. SemEval-2021 task 10: Source-free domain adaptation for semantic processing. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval)*.

Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. SemEval 2018 task 6: Parsing time normalizations. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 88–96, New Orleans, Louisiana. Association for Computational Linguistics.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*, abs/1907.11692.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. 2016. The variational fair autoencoder. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. *Snippext: Semi-Supervised Opinion Mining with Augmented Data*, page 617–628. Association for Computing Machinery, New York, NY, USA.

Patrice Y. Simard, Dave Steinkraus, and John C. Platt. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, page 958, USA. IEEE Computer Society.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

A Appendix

A.1 Entity Candidates Pool

Second-Of-Minute: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15,16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60.

Day-Of-Month: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

This: today, these, this, the, now, current, These, This, The, Current.

Minute-Of-Hour: 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60.

Year: 1970, 1971, 1973, 1974, 1975, 1980, 1981, 1982, 1984, 1990, 1992, 1995, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2020.

Month-Of-Year: January, February, March, April, May, June, July, August, September, October, November, December, Jan, Feb, Mar, Apr, Aug, Sept, Sep, Oct, Nov, Dec, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, may.

Next: later, future, following, next, coming, upcoming, Following.

Hour-Of-Day: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 01, 02, 03, 04, 05, 06, 07, 08, 09.

Time-Zone: ART, CT, EGT, EST, MART, MMT, NT, TOT, WIT.

Two-Digit-Year: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, tens, Tens.

Calendar-Interval: minute, minutes, day, days, daily, today, eve, week, weekly, weeks, months, month, monthly, quarter, quarterly, year, years, annual, annually, Daily, Eve, Month, Monthly, Year, Annual.

Modifier: more than, approx, less than, start, mid, middle, end, over, early, around, late, older, financial, fiscal, nearly, longer than, almost, at least, or so, about, beginning, More than, Approx, Less than, Mid, Middle, End, Over, Early, Around, Late, Nearly, At least, About, Beginning.

Last: before, last, latest, previously, recent, recently, the past, ever, previous, past, earlier, pre, Before, Last, Recent, Ever, Previous, Past, Earlier, Pre.

Between: from, since, until, between, From, Since, Until, Between.

Day-Of-Week: Tuesday, Wednesday, Mon, Tues, Tue, Wed, Sat, Sun.

Period: period, periods, week, months, minute, year, term, day, time, years, second, moment, minutes, long-term, decades, decade, short-term, month, weeks, days, Year, Term, Time, Second, Short-term, Month.

Part-Of-Day: night, Noon.

Before: previously, prior, before, ago, pre, by, earlier, next, Prior, Before, Ago, Pre, By, Earlier.

NthFromStart: second, first, fourth, third, seventh, Second, 3rd, 5th, 7th, 25th, 47th, 75th.

After: after, from, later, post, After, From, Post.

Season-Of-Year: winter, spring, summer, fall, season, autumn, Summer, Fall, Season.

AMPM-Of-Day: pm, am, PM, AM.

A.2 Hyperparameters

Hyperparameter	Value
number of examples from the test set used for self-training	3000
maximum number of self-training iterations	30
actual number of self-training iterations	2
self-training threshold (τ)	0.95
maximum sequence length	128
batch size	32
epochs	10
learning rate	5e-5
learning rate schedule type	linear
learning rate warm up steps	0
weight decay	0.0
maximum gradient norm	1.0

Table 4: Hyperparameters for negation detection system

Hyperparameter	Value
number of active learning iterations (i)	5
number of sentences to annotate at each active learning iteration (k)	32
number of new sequence to augment for each annotated time entity (n)	5
maximum sequence length	271
batch size	32
epochs	8
learning rate	3e-5
learning rate schedule type	linear
learning rate warm up steps	0
weight decay	0.0
maximum gradient norm	1.0

Table 5: Hyperparameters for time expression recognition system.

A.3 Manual Annotation Performance

Type	F	P	R	# in gold annotations	# in our annotations
I-Calendar-Interval	0.000	0.000	0.000	0	5
I-Last	0.000	0.000	0.000	2	5
I-Between	0.000	0.000	0.000	1	0
I-Modifier	0.000	0.000	0.000	1	4
B-This	0.154	0.333	0.100	10	3
B-Before	0.222	0.250	0.200	5	4
B-Union	0.250	0.143	1.000	1	7
B-Period	0.400	0.308	0.571	7	13
B-After	0.500	1.000	0.333	9	3
I-Frequency	0.500	1.000	0.333	3	1
B-Modifier	0.519	0.389	0.778	9	18
I-Period	0.588	1.000	0.417	12	5
B-Last	0.615	0.800	0.500	8	5
B-Calendar-Interval	0.632	0.529	0.783	23	34
B-Intersection	0.667	1.000	0.500	2	1
B-Frequency	0.750	0.750	0.750	4	4
I-Number	0.769	1.000	0.625	8	5
B-Season-Of-Year	0.792	0.864	0.731	52	44
B-NthFromStart	0.800	0.667	1.000	2	3
B-Between	0.831	0.750	0.931	29	36
I-Season-Of-Year	0.896	0.972	0.831	83	71
B-Number	0.909	0.909	0.909	11	11
O	0.997	0.997	0.998	12627	12632
B-Year	1.000	1.000	1.000	36	36
B-Month-Of-Year	1.000	1.000	1.000	57	57
B-Part-Of-Day	1.000	1.000	1.000	1	1
B-Two-Digit-Year	1.000	1.000	1.000	4	4
B-Next	1.000	1.000	1.000	1	1

Table 6: Performance of each annotated entity types. Please note that when the number in gold annotation is 0, it means that we annotate this entity type, but it does not appear in the gold annotations (test labels).