To Schedule or not to Schedule: Extracting Task Specific Temporal Entities and Associated Negation Constraints

Barun Patra, Pamela Bhattacharya, Chala Fufa, Charles Lee
Microsoft
{bapatra, pamelabh, chfufa, charlle}@microsoft.com

Abstract

State of the art research for date-time[1] entity extraction from text is task agnostic. Consequently, while the methods proposed in literature perform well for generic date-time extraction from texts, they don't fare as well on task specific date-time entity extraction where only a subset of the date-time entities present in the text are pertinent to solving the task. Furthermore, some tasks require identifying negation constraints associated with the date-time entities to correctly reason over time. We showcase a novel model for extracting task-specific date-time entities along with their negation constraints. We show the efficacy of our method on the task of date-time understanding in the context of scheduling meetings for an email-based digital AI scheduling assistant. Our method achieves an absolute gain of 19% f-score points compared to baseline methods in detecting the date-time entities relevant to scheduling meetings and a 4% improvement over baseline methods for detecting negation constraints over date-time entities.

1 Introduction

Temporal entity extraction and normalization is an important aspect of Natural Language Processing (Alonso et al., 2011; Campos et al., 2014). There has been a substantial body of work on the task and there exist numerous well performing publicly available models for identifying and normalizing temporal entities (Strötgen and Gertz, 2010; Chang and Manning, 2012; Zhong and Cambria, 2018).

There exist however a growing number of NLP applications which require extraction of only a relevant subset of time entities that are useful for solving specific problems within a larger body of text. Examples of such tasks include understanding search queries ("Find me all emails sent by April between May 11^{th} and May 21^{st}"), Goal Oriented Dialogue Systems ("Deliver George Orwell's 1984 by next week.", "Send the "FY 2020 Budget" to Watson Monday morning.") etc. Using the temporal entity extraction models for these tasks is insufficient, since they fail to disambiguate between general date-time entities and entities necessary to solve the task.

In this paper, we address the task of recognizing date-time entities required by an AI scheduling assistant for correctly scheduling meetings. Cortana from Microsoft Scheduler, Clara from Clara Labs and Amy from X.ai are examples of such email based digital assistants for scheduling meetings. For such systems, a user organizing the meeting adds the digital assistant as a recipient in an email with other attendees and delegates the task of scheduling to the digital assistant in natural language. For the assistant to correctly schedule the meeting, it must correctly extract the date-time entities expressed by the user in the email to indicate the times they want the meeting scheduled, as well as the times that do not work for them. The verbose nature of emails often exacerbates the difficulty of identifying relevant date-time entities; since the number of distractor (i.e valid date-time entities not pertinent to the task) tend to increase (Eg: In Fig. 1 "today" serves as a distractor entity).

To this end, we present SHERLOCK: **Sc**ch**H**eduling **E**ntity **R**ecovery by **LO**oking at **C**ontextual **K**nowledge, a novel model for detecting relevant date-time entities in the context of scheduling as well as identifying the entities associated with a negation constraint. SHERLOCK comprises of 3 modules for identifying the relevant entities as well as negation constraints associated with them:

[1] We use date-time entities, date entities, time entities and temporal entities interchangeably to denote entities associated with dates and/ or times.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 8445–8455,
November 16–20, 2020.

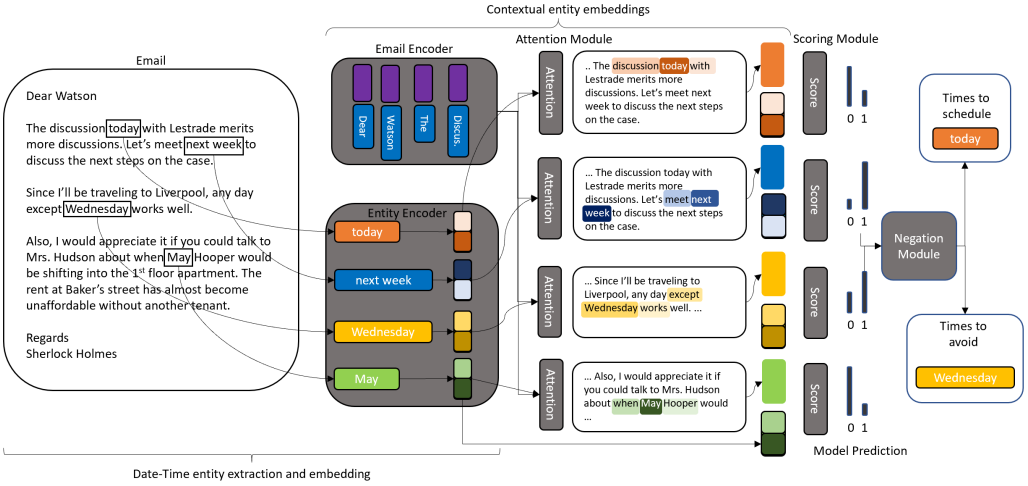

Figure 1: The 3 modules of SHERLOCK: First a high recall rule based extractor generates the potential entities. The neural module then takes the email and the entities and generates scores for each entity. Only the relevant entities are passed to the final negation module to detect times to schedule and times to avoid.

Date-Time Extractor: A high recall date-time entity extractor to identify all date-time entities in an email

Entity Relevance Scorer: A neural model to classify each of the extracted entities as being relevant to scheduling or not by considering the context presented in the email.

Negation Detector: A negation module to identify if there exists a negation constraint associated with each of the extracted relevant entities.

Fig. 1 illustrates each module: the entity extractor extracts "today", "next week", "Wednesday" and "May". Each of these entities is scored by the neural module, and only "next week" and "Wednesday" are identified as being relevant to scheduling. Finally, the negation module identifies that "Wednesday" has a negation constraint. While SHERLOCK focuses on the task of scheduling, we believe that a similar approach can be used to tackle the problem of extracting relevant date-time entities from documents for other tasks.

The contributions of this paper are as follows:

Task specific date-time extractor: A novel method for combining conventional high recall rule-based model with a novel neural model for incorporating contextual information to identify relevant date-time entities for the task at hand.

Identifying negation constraints for temporal entities: A heuristic negation module that helps identify negation constraints associated with time entities in the context of scheduling meetings. To the best of our knowledge, prior to this work, negation constraints associated with time-entity extraction have not been studied before.

We first present our proposed method for extracting time entities relevant to the task of scheduling a meeting in (§2). Next, we describe our approach for identifying negation constraints associated with extracted entities in (§3). In (§4), we describe our experimental setup and baselines. We discuss the results in (§5) and show that SHERLOCK helps improve performance both on the task of identifying relevant entities as well as identifying negation constraints. We then present the related work in (§6), and finally conclude in (§7).

2 Contextual Date-Time Extraction

In order to correctly extract relevant temporal entities in the context of scheduling meetings from an email, we first extract potential entities using an off-the-shelf date-time entity extractor. Both the email and the extracted entities are then encoded using neural modules. For each extracted entity, we then generate a context embedding using the encoded email and the encoded entity. Both the contextual and encoded entity embedding are then used to predict if an entity is relevant or not. We describe each component in detail below:

2.1 Entity Extraction and Encoding

Given an email $\boldsymbol{X} = \{w_1 \cdots w_n\}$, we first use a rule-based tagger for extracting potential date-time

entities from an email. Specifically, we use LUIS[2] (Williams et al., 2015) for extracting the entities. The model is recall heavy and identifies potential time utterances (Eg: in Figure 1, LUIS detects "today", "next week", "wednesday", "may"). We denote the extracted entities as $\mathcal{E} = \{e_1 \cdots e_m\}$, where $e_i = \{e_{i,1} \cdots e_{i,l_i}\}$ represents the i^{th} entity and l_i denotes the length of e_i.

For each entity e_i, we generate an embedding $u_{e_i} \in \mathbb{R}^{d_e}$ (where d_e denotes the entity embedding dimension) as follows:

$$\begin{aligned} t_{i,j} &= \texttt{LookUp}(e_{i,j}) \\ r_{i,j} &= \texttt{CharEncoder}(e_{i,j}) \\ h_{i,j} &= [r_{i,j}; t_{i,j}] \\ u_{e_i} &= \texttt{Seq2SeqEncoder}(h_{i,1} \cdots h_{i,l_i}) \end{aligned} \tag{1}$$

In Equation (1), $t_{i,j}$ denotes the word level embedding of the j^{th} word if the i^{th} entity ($e_{i,j}$). As is standard practice, OOV words all share a common word embedding, while other entities encountered during training are represented by a learnt vector. We also augment this with an embedding from a character level encoder. $r_{i,j}$ denotes the word level embedding obtained by passing $e_{i,j}$[3] through a character level encoder, which allows the model to represent OOV entities. The two embeddings are concatenated, and then passed through another `Seq2SeqEncoder` model (any Sequence-to-Sequence encoder (Sutskever et al., 2014)) to get the final entity encoding u_{e_i}[4].

2.2 Contextual Entity Embeddings

From Figure 1, we can observe that it is clear from context that "May" is not a time entity, and "today" is not an entity relevant to scheduling. We want to capture this contextual information for each entity. To do so, we first encode the email as follows:

$$(v_{w_1} \cdots v_{w_n}) = \texttt{Seq2SeqEncoder}(w_1 \cdots w_n) \tag{2}$$

In Equation (2), v_{w_i} denotes the embedding for the i^{th} word of the email $\boldsymbol{X}$. d_w here denotes the embedding size for the email embedding.

Once we have the email embeddings, we then compute the contextual embedding for each entity using an attention mechanism (Bahdanau et al., 2014). For entity e_i, given the entity embedding u_{e_i}, and the email embeddings $(v_{w_1} \cdots w_{w_n}), v_{w_i} \in \mathbb{R}^{d_w}$, the contextual embedding $c_{e_i} \in \mathbb{R}^{d_w}$ is obtained as follows:

$$\begin{aligned} a_{w_j} &= Av_{w_j} + b \\ b_{w_j} &= \texttt{tanh}(a_{w_j} + u_{e_i}) \\ logit_{w_j} &= Bb_{w_j} + d \\ \alpha w_j &= \texttt{softmax}(logit_{w_j}) \\ c_{e_i} &= \sum_{w' \in \{w_1 \cdots w_n\}} \alpha_{w'} v_{w'} \end{aligned} \tag{3}$$

Where $A \in \mathbb{R}^{d_e \times d_w}, b \in \mathbb{R}^{d_e}, B \in \mathbb{R}^{d_e \times 1}, d \in \mathbb{R}$ are learned parameters. The final entity embedding (f_{e_i}) is the concatenation of the entity embedding and the contextual embedding. Finally, for each entity, we generate a probability score to indicate if an entity is relevant or not.

$$\begin{aligned} f_{e_i} &= [u_{e_i}; c_{e_i}] \\ s_{e_i} &= \sigma(M f_{e_i} + g) \end{aligned} \tag{4}$$

Where $M \in \mathbb{R}^{(d_e + d_w) \times 1}, g \in \mathbb{R}$ are learned parameters, and σ indicates the sigmoid function.

2.3 Learning

Given the entities that are relevant to scheduling $\mathcal{Y} \subseteq \mathcal{E}$ (Eg: "next week" and "Wednesday" in Figure 1), we train the model with a scoring loss as follows:

$$\mathcal{L}_s = -(\sum_{e \in \mathcal{Y}} \log(s_e) + \sum_{e \in \mathcal{E} \setminus \mathcal{Y}} \log(1 - s_e)) \tag{5}$$

Similar to (Ruder, 2017; Gehrmann et al., 2018; Li et al., 2018), we find that augmenting the learning with a related auxiliary task helps improve performance. In this case, a simple related auxiliary function is the task of sequence tagging. Specifically, given the email $\boldsymbol{X}$, and the relevant entities $\mathcal{Y}$, we tag the location of each entity with I-Time tag, and every other token with an O tag. Let the generated tags be $\boldsymbol{z} = (z_1 \cdots z_n)$ and let C denote the set of possible tagging labels (in our case 2: {I-Time, O}) We then train a standard CRF for

[2] https://www.luis.ai/home

[3] Technically the embeddings associated with characters of $e_{i,j}$ are passed to the character level encoder

[4] For a unidirectional encoder, the final hidden state is used as the embedding. The concatenation of the forward and backward hidden states is used for a bidirectional encoder

tagging as follows:

$$
\begin{aligned}
g_i &= Pv_i + q \\
\texttt{score}(\boldsymbol{X}, \boldsymbol{z}) &= \sum_{i=2}^{n} T_{z_i, z_{i-1}} + \sum_{i=1}^{n} g_{i,z_i} \\
p(\boldsymbol{z}|\boldsymbol{X}) &= \frac{e^{\texttt{score}(\boldsymbol{X}, \boldsymbol{z})}}{\sum_{\boldsymbol{z}' \in \mathcal{Z}} e^{\texttt{score}(\boldsymbol{X}, \boldsymbol{z}')}} \\
\mathcal{L}_t &= -(\texttt{score}(\boldsymbol{X}, \boldsymbol{z}) - \\
&\quad \log(\sum_{\boldsymbol{z}' \in \mathcal{Z}} e^{\texttt{score}(\boldsymbol{X}, \boldsymbol{z}')}))
\end{aligned}
\tag{6}
$$

Where $P \in \mathbb{R}^{d_w \times |C|}, q \in \mathbb{R}^{|C|}$ are trainable parameters, $T \in \mathbb{R}^{|C| \times |C|}$ is the transition matrix and $\mathcal{Z}$ is the set of all possible sequence labels.

The final loss that we optimize for is

$$
\mathcal{L}_{final} = \gamma \mathcal{L}_s + (1 - \gamma)\mathcal{L}_t \tag{7}
$$

Where γ balances between the two loss functions.

2.4 Choosing the Prediction Threshold

In order to find the threshold for classifying the positive class (i.e t such that $e_i = 1$ if $s_{e_i} > t$), we compute the F1 score on the validation set using a grid of thresholds[5], and choose the threshold maximizing the F1 score.

3 Identifying Negation Constraints

For a Scheduling Assistant to be able to correctly schedule meetings, understanding negations is crucial; otherwise it can lead to an unsatisfactory user experience (E.g.: In Figure 1, the meeting being scheduled on Wednesday would be a frustrating experience for the organizer Sherlock). Only about 10% of scheduling requests in our dataset have negation constraints. Building a model directly for the task did not show promising results from our preliminary experiments. We hypothesize this was due to the small volume of the data as well as the lack of good quality supervised data. Consequently, to find negated time-entities, we adopt the approach of first finding the negation scope. If an entity occurs inside the negation scope, we mark it to be negated.

In order to find the negation scope, we build on the approach proposed in Rosenberg (2013). We first find the negation cue ("except" in Figure 1). To find the negation cue, we first tokenize the email into sentences. For each sentence, we try fo find if cue from a set of negating cues (Appendix A) occurs in the sentence.

After finding the negation cue, we identify the POS tag of the negating cue (Prep. for "except"). Given the POS tag and the negation cue, we trigger a set of heuristics to identify the negation scope. Most heuristics work by identifying the negation cue from the dependency parse of the sentence as well as the governor of the negating word. Generating the narrow scope of negation (i.e. not containing the subject) then involves identifying the constituent from the constituency parse that contains both the negation cue and the governor word ("any day except Wednesday", see Figure 2). This constituent is considered to be the candidate narrow scope, and usually, the part following the cue is considered to be the narrow scope.

For some cases, the narrow negation scope is not enough to identify the time entity being negated. Consider the second example from Figure 2:

Example: *Next week does not work Watson.*
Narrow: *Next week does not [work Watson].*
Wide: *[Next week] [does] not [work Watson].*

For this case, the narrow scope is not enough to identify the entity being negated ("next week"). To find the wide scope, the heuristics leveraging the dependency path starting from the governor word are used. The main idea is to find the subject associated with the governor node, and extract that as the wide scope ("Next week"). Following the guidelines set by Morante and Daelemans (2012), we also include the aux dependency node in the wide scope ("does").

We also expand the heuristic set presented in Rosenberg (2013), adding the following rules:

- If a Noun Phrase (NP) acting as an adverbial modifier acts as a subject to the governor, we include it in the wide scope (Figure 2)
- If a NP exists as a subject of a passive clause, we include it in the wide scope, as well as the passive auxiliary associated with it.
- A Prepositional Phrase (PP) acting as a subject to the governor is included in the wide scope.
- For the narrow scope, we prune out the subtree that exists as an object an adverbial clause relation (advcl) headed by the governor node.

Due to space constraints, we include examples for the above in Appendix B.

[5] We use the precision_recall_curve provided by sklearn (Buitinck et al., 2013)

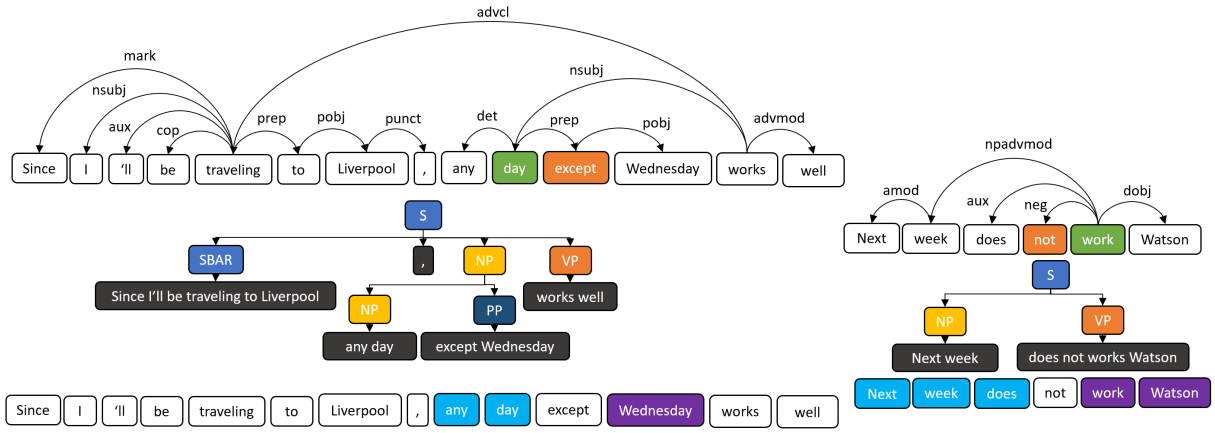

Figure 2: The negation extraction model. Orange indicates the negation cue, Green denotes the governing node. Purple denotes the narrow scope, and Light Blue denotes the wide scope.

After obtaining the narrow and wide scopes, we check if any entities are found in the narrow scope. If found, those entities are scored negated. If no entities are found in the narrow scope, we then check the wide scope to find negated entities.

Finally, for some cases, we also use domain specific cues that imply a non-availability (For example, in "Dr. John out of office on Monday.", "out of office" implies an unavailability to meet.) When such implied negation cues are encountered, we default to a custom heuristic which marks any entity occurring within the sentence containing the cue word as a negation.

4 Experimental Setup

We first show the effectiveness of our proposed entity scoring method of incorporating context for improving temporal entity extraction on the TempEval-3 dataset (UzZaman et al., 2013) (§4.1). We then show the efficacy of SHERLOCK for the task of extracting the correct temporal entities relevant for the context of scheduling, a task for which context becomes substantially more important (§4.2). Finally, we show that SHERLOCK's negation module outperforms baseline methods on the task of identifying the entities with negation constraints (§4.3). All our models have been implemented using the AllenNLP framework (Gardner et al., 2017). The hyperparameters for all the experiments can be found in Appendix C and D.

4.1 TempEval-2013

4.1.1 Dataset

We use the TimeBank dataset (Pustejovsky et al., 2003) which serves as the benchmark dataset for the TempEval series. The dataset consists of 256 documents, comprising of 95,391 tokens and 1,822 TimeEx entities for training and validation purposes, and 20 documents (6,375 tokens, 138 TimeEx) for serving as the test set.

4.1.2 Baseline Models

We show the performance of augmenting 3 rule-based models with our proposed model. Specifically, we consider SUTime (Chang and Manning, 2012), HeidelTime (Strötgen and Gertz, 2010) and Syntime (Zhong et al., 2017) as the rule-based extractors. We also compare against UWTime (Lee et al., 2014), a learning based model.

4.1.3 Evaluation

We use the official TempEval-3 scoring script and report the standard metrics. Specifically, we report the detection precision, recall and F1 with the relaxed and strict metrics. A gold mention is considered for the relaxed metric if any of the output candidates overlap with it and for the strict case, an exact string match is considered.

4.2 Date-Time extraction for Scheduling

This task aims at extracting the date-time entities necessary for the Scheduling Agent to correctly schedule the meeting. The task necessarily needs the model to incorporate context for making the correct prediction (E.g.: In Figure 1, "today" is a valid date-time entity, but not relevant for scheduling, while "May" refers to a person.)

4.2.1 Dataset

We use an internal scheduling dataset for training and evaluating the models. The dataset consists

of emails and annotated times to schedule. The training and validation set consists of 44,214 emails (4,589,631 tokens, and 48083 entities), while the test set consists of 4914 emails (519,021 tokens, 5233 entities).

4.2.2 Baseline Models

We compare the performance of our model against SUTime, HidelTime and LUIS. We use LUIS as our base date-time extractor since it provides a much larger coverage for date-time entities [6].

4.2.3 Evaluation

We use the Strict F1 measure to compare the performance of the different models proposed.

4.3 Negation Detection

Finally, we compare the performance of our proposed model on the task of negation extraction.

4.3.1 Dataset

We use an internal dataset for comparing different models on the task of negation extraction. The dataset consists of 1253 emails for which time-entities that are relevant to scheduling are selected, and those that are a part of a negation constraint are marked as negated entities. There exist 3231 time-entities, of which 1589 are negated entities.

4.3.2 Baselines

We compare our proposed method against a naive heuristic method as well as a neural model trained on a publicly available negation scope detection dataset.

Heuristic: A naive heuristic model. If a negation cue is identified in a sentence, the model predicts that all entities in that sentence are negated.

NegNN: We use a NegNN model (Fancellu et al., 2016), modified to use BERT contextual embeddings and trained on the *SEM2012 Shared task (Morante and Blanco, 2012). The training, development and test sets are a collection of stories from Conan Doyle's *Sherlock Holmes*, with the cue and scope annotated. An entity is considered negated if it is a part of a negated scope, as predicted by the model. The performance of the modified NegNN model on the *SEM2012 Task can be found in Appendix E.

[6]For example, LUIS recognizes military time ("1530"), and has a much larger coverage for holidays

4.3.3 Evaluation

We measure the performance of different models by comparing the predicted set of negated entities and the gold labels for the entities. If the model makes a mistake (i.e. it predicts an entity to be negated, when it's not), that's considered a false positive. Likewise, any negated entities missed by the model contribute to the false negatives. We thus report the precision, recall and F1 score.

5 Results and Analysis

5.1 TempEval-2013

Model	Strict			Relaxed		
	Pre.	Rec.	F1	Pre.	Rec.	F1
SUTime	80.0	81.2	80.6	90.0	91.3	90.7
SUTime(+)	85.9	79.7	82.7	93.6	87.0	90.2
HeidelTime	83.9	79.7	81.7	93.1	88.4	90.7
HeidelTime(+)	84.6	79.7	82.1	93.1	87.7	90.3
Syntime	91.4	92.7	92.1	94.3	95.7	95.0
Syntime(+)	92.7	92.0	92.4	94.9	94.2	94.6
UWTime	84.6	83.4	84.0	92.8	91.5	92.1

Table 1: Performance on TempEval Dataset. Models with (+) indicate that the base extractor is augmented with the entity scoring module (Scale: 0-100)

Table 1 shows the performance of SHERLOCK's entity scoring module on the TempEval-2013 dataset. Note that SHERLOCK is limited by the recall of the base rule-based extractor[7]. We observe that augmenting the rule-based model with SHERLOCK improves the precision for all three cases without a substantial drop in recall. Furthermore, the precision obtained for all the augmented models compares favorably with UWTime.

5.2 Date-Time extraction for Scheduling

Model	Precision	Recall	F1
LUIS	0.38	0.98	0.54
SUTime	0.59	0.79	0.68
HidelTime	0.66	0.86	0.75
SHERLOCK - $\mathcal{L}_t$	0.91	0.96	0.93
SHERLOCK	0.91	0.98	0.94

Table 2: Performance on Date-Time Extraction for Scheduling. SHERLOCK - $\mathcal{L}_t$ denotes the SHERLOCK model without the tagging loss (Scale: 0 - 1)

Table 2 shows the performance of SHERLOCK for the Scheduling related date-time extraction task.

[7]The model only scores the predictions of the base extractor

As can be seen, being able to incorporate context yields a substantial improvement over the baseline methods.

We also observed that incorporating the tagging loss $\mathcal{L}_t$ helped improve performance (SHERLOCK vs SHERLOCK - $\mathcal{L}_t$). On investigating further, we observed that the attention weights associated with an entity for a model trained with $\mathcal{L}_t$ concentrated much better around the position of the entity in the email than for the model without it. To see why that is advantageous, consider the following example:

"Let's schedule for tomorrow. Next month, I plan on taking up Mr Baskerville's case"

Here, the model without $\mathcal{L}_t$ generates high attention weights for embeddings associated with "tomorrow", since the localization of the attention weights is much more spread out. Consequently, it also uses the embeddings associated with "tomorrow" for predicting the label of "next month", and hence, predicts it to be relevant to scheduling when it is not. Due to space constraints, we include our localization experiments in Appendix F.

5.3 Negation Detection

Model	Precision	Recall	F1
NegNN	0.73	0.13	0.22
Heuristic	0.78	0.63	0.70
SHERLOCK	0.91	0.62	0.74

Table 3: Negation Performance (Scale: 0 - 1)

Category	Model	Precision	Recall	F1
Explicit	NegNN	0.83	0.25	0.39
	Heuristic	0.76	0.86	0.81
	SHERLOCK	0.94	0.87	0.90
Implied	NegNN	0.23	0.01	0.03
	Heuristic	0.83	0.42	0.56
	SHERLOCK	0.87	0.40	0.54

Table 4: Explicit vs implied negations (Scale: 0 - 1)

Table 3 shows the performance of SHERLOCK compared to the baseline methods. We hypothesize the reason why SHERLOCK and the simple heuristic model outperform the neural baseline is two-fold: the neural negation model was trained on a dataset of Sherlock Holmes stories and consequently does not adapt well when used for negation extraction for emails; and that the neural model has no notion of implied negations.

To test this hypothesis, we split the negations into two categories: explicit negations (defined as a negation where the cue is one of the explicit negation cues), and the case wherein the negation is implied (any case that was not explicit was deemed implied). 50% of emails in the negation dataset contained explicit negations only, 48% contained implied negations only and 2% contained both.

Table 4 shows the performance of SHERLOCK and the baselines for both the explicit negation and the implied negation cases. Unsurprisingly, we see that both the baselines as well as SHERLOCK perform better on explicit negations than they do on implied negations. However, the gains observed by both the heuristic model and SHERLOCK substantially outperform NegNN, with SHERLOCK substantially outperforming the heuristic. Examples 1 and 2 in Table 5 give qualitative examples of where SHERLOCK outperforms the heuristic.

The primary source of errors for detecting implied negations is from failing to identify the correct cue. Since heuristics for implied negations are more heavily focused on precision, the absence of negation cues results in the model not detecting the implied negation, which in turn negatively impacts the recall. Examples 3, 4 and 5 in Table 5 show cases where the cue is not present in the heuristic set of implied cues.

For explicit negations, one source of errors is due to entity co-referencing. Consider Example 6: the negated time instance Tuesday is referenced as "then" and hence the negation scope "then" is insufficient to identify the correct negated entity. A few errors also stem from inherent ambiguity: in Example 7, the request can either be interpreted as being for anytime next week except Thursday 10am, or for 10 am on all days except Thursday. Finally, we also observe errors due to double negations (Example 8) and due to incorrect constituency and dependency parses.

6 Related Work

Existing approaches for time expression extraction can be categorized into rule-based methods and learning-based methods.

Rule-based Methods Rule-based methods like HeidelTime, and SUTime mainly handcraft deterministic rules to identify time expressions. TempEx and GUTime use both hand-crafted rules and machine-learnt rules to resolve time expressions (Mani and Wilson, 2000; Verhagen et al., 2005; Blamey et al., 2013). HeidelTime manually designs rules with time resources to recognize time

Idx	Example	Heuristic	SHERLOCK	Correct
1	Mycroft cannot do Monday but Tuesday should work fine.	[Monday, Tuesday]	[Monday]	[Monday]
2	If Watson is not busy, Wednesday also works.	[Wednesday]	[]	[]
3	I'm slammed on Thursday. - Lestrade	[]	[]	[Thursday]
4	I am out of town on Wednesday Irene but Thursday might work.	[]	[]	[Wednesday]
5	I am completely booked with appointments on Thursday Sherlock. – Watson	[]	[]	[Thursday]
6	Mr. Holmes, my trip's on Tuesday. I really can't meet then.	[]	[]	[Tuesday]
7	Let's just meet next week any day except Thursday at 10:00 am. - Holmes	[next week, Thursday at 10:00 am]	[Thursday]	[Thursday at 10:00 am]
8	Next week would not be possible, except on Friday.	[Next week, Friday]	[Next week, Friday]	[next week]

Table 5: Examples of SHERLOCK's Negation Model's predictions and errors

expressions (Strötgen and Gertz, 2010). SUTime designs deterministic rules at three levels (i.e., individual word level, chunk level, and time expression level) for time expression recognition (Chang and Manning, 2012). A recent type-based time tagger, SynTime, designs general heuristic rules with a token type system to recognize time expressions (Zhong et al., 2017). TOMN (Zhong and Cambria, 2018) uses the token regular expressions, similar to SUTime (Chang and Manning, 2012) and SynTime (Zhong et al., 2017), and further groups them into three token types, similar to SynTime. TOMN also leverages statistical information from entire corpus to improve the precisions and alleviate the deterministic role of deterministic and heuristic rules.

Learning-based Method Learning-based methods in TempEval series mainly extract features from text (e.g., character features, word features, syntactic features, and semantic features), and on the features apply statistical models (e.g., CRFs) to model time expressions (Bethard, 2013; Filannino et al., 2013; Llorens et al., 2010; UzZaman and Allen, 2010). Besides the standard methods, (Angeli et al., 2012; Angeli and Uszkoreit, 2013) exploit an EM-style approach with compositional grammar to learn latent time parsers. (Lee et al., 2014) leverage a learnt CCG (Steedman, 1996) parser and define a lexicon with linguistic context to model time expressions, using the loose structure information by grouping the constituent words of time expression under three token types.

Negation Scope Detection: Most negation detection research has focused in the Bio-Medical domain (Mehrabi et al., 2015; Agarwal and Yu, 2010). Non Bio-Medical text related negation detection tasks usually involve learning supervised classifiers over hand-crafted features leveraging syntactic structure (constituency and dependency parses) (Velldal et al., 2012; Lapponi et al., 2012; Chowdhury and Mahbub, 2012; White, 2012; Abu-Jbara and Radev, 2012). The current state of the art learned method uses a Neural BiLSTM-CRF model (Fancellu et al., 2016). However, the corpus available for negation detection is on Sherlock Holmes stories (*SEM2012 Shared task (Morante and Daelemans, 2012)), and consequently, as shown in this work, do not adapt well on language used in other document styles (like emails). In this work, we built over the work of (Rosenberg, 2013), who develop linguistic rules over constituency and dependency parses to identify negation scopes. The primary advantage of leveraging their work is that it is not strongly tied to the *SEM 2012 dataset, and we found this to generalize better.

Finally, there has been some work on directly training a model to extract entities and associated negation constraints (Bhatia et al., 2019). However, these works usually assume the availability of good quality annotated negated entities. Given enough annotated data, exploring this direction would be an interesting line of future work.

7 Conclusion

In this paper, we presented a novel model that leverages conventional high recall rule-based models and neural models for utilizing contextual informa-

tion for identifying task relevant temporal entities. Our proposed model, when used in conjunction with 3 different rule-based models, achieves substantial precision gains for all of them without suffering from a huge recall drop. Further, the model substantially outperforms baseline methods for the task of identifying relevant date-time entities for the task of scheduling a meeting.

We also presented a novel approach for identifying the negation constraints of date-time entities. Identifying the negation constraints associated with date-time entities correctly is necessary for the task of scheduling. We showed that the existing neural approaches for detecting negation scopes do not transfer well, and that our proposed model based on heuristics defined over constituency and dependency parses achieves strong performance gains, especially for the case of explicit negations.

Acknowledgements

We would like to thank Vishwas Suryanarayanan for his valuable comments and discussions. We would also like to thank the anonymous reviewers, whose valuable comments and suggestions helped shape the paper into its current form.

References

Amjad Abu-Jbara and Dragomir Radev. 2012. Umichigan: A conditional random field model for resolving the scope of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 328–334. Association for Computational Linguistics.

Shashank Agarwal and Hong Yu. 2010. Biomedical negation scope detection with conditional random fields. *Journal of the American medical informatics association*, 17(6):696–701.

Omar Alonso, Jannik Strötgen, Ricardo A Baeza-Yates, and Michael Gertz. 2011. Temporal information retrieval: Challenges and opportunities. *Twaw*, 11:1–8.

Gabor Angeli, Christopher D. Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL.*

Gabor Angeli and Jakob Uszkoreit. 2013. Language-independent discriminative parsing of temporal expressions. In *In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

David Arthur and Sergei Vassilvitskii. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

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

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval. In *In Proceedings of the 7th International Workshop on Semantic Evaluation.*

Parminder Bhatia, Busra Celikkaya, and Mohammed Khalilia. 2019. Joint entity extraction and assertion detection for clinical text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 954–959, Florence, Italy. Association for Computational Linguistics.

Benjamin Blamey, Tom Crick, and Giles Oatley. 2013. 'the first day of summer': Parsing temporal expressions with distributed semantics. *Research and Development in Intelligent Systems*, pages 389–402.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Ricardo Campos, Gaël Dias, Alípio M Jorge, and Adam Jatowt. 2014. Survey of temporal information retrieval and related applications. *ACM Computing Surveys (CSUR)*, 47(2):1–41.

Angel X. Chang and Christopher Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3735–3740.

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. *arXiv preprint arXiv:1406.1078*.

Md Chowdhury and Faisal Mahbub. 2012. Fbk: Exploiting phrasal and contextual clues for negation scope detection. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 340–346. Association for Computational Linguistics.

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*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, pages 495–504.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 53–57.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Emanuele Lapponi, Erik Velldal, Lilja Øvrelid, and Jonathon Read. 2012. Uio 2: sequence-labeling negation using dependency features. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 319–327. Association for Computational Linguistics.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1437–1447.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium. Association for Computational Linguistics.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *ACL '00 Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 69–76.

Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C Max Schmidt, Hongfang Liu, et al. 2015. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213–219.

Roser Morante and Eduardo Blanco. 2012. * sem 2012 shared task: Resolving the scope and focus of negation. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 265–274.

Roser Morante and Walter Daelemans. 2012. Conandoyle-neg: Annotation of negation in conan doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, Istanbul*, pages 1563–1568.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK.

Sabine Rosenberg. 2013. *Negation triggers and their scope*. Ph.D. thesis, Concordia University.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Mark Steedman. 1996. *Surface Structure and Interpretation*.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Naushad UzZaman and James Allen. 2010. Trips and trios system for tempeval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 276–283.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics*, pages 1–9.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational linguistics*, 38(2):369–410.

Marc Verhagen, Inderjeet Mani, Roser Sauri, Jessica Littman, Robert Knippen, Seok B. Jang, Anna Rumshisky, John Phillips, and James Pustejovsky. 2005. Automating temporal annotation with tarsqi. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 81–84.

James Paul White. 2012. Uwashington: Negation resolution using machine learning methods. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 335–339. Association for Computational Linguistics.

Jason D Williams, Eslam Kamal, Mokhtar Ashour, Hani Amr, Jessica Miller, and Geoff Zweig. 2015. Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (luis). In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 159–161.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Xiaoshi Zhong and Erik Cambria. 2018. Time expression recognition using a constituent-based tagging scheme. In *WWW '18*, pages 983–992.

Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time expression analysis and recognition using syntactic token types and general heuristic rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 420–429.