# Extracting Person Names from User Generated Text: Named-Entity Recognition for Combating Human Trafficking

**Yifei Li**
**Pratheeksha Nair**
**Kellin Pelrine**
**Reihaneh Rabbany**
School of Computer Science
McGill University
Mila – Quebec AI Institute

## Abstract

Online escort advertisement websites are widely used for advertising victims of human trafficking. Domain experts agree that advertising multiple people in the same ad is a strong indicator of trafficking. Thus, extracting person names from the text of these ads can provide valuable clues for further analysis. However, Named-Entity Recognition (NER) on escort ads is challenging because the text can be noisy, colloquial and often lacking proper grammar and punctuation. Most existing state-of-the-art NER models fail to demonstrate satisfactory performance in this task. In this paper, we propose NEAT (Name Extraction Against Trafficking) for extracting person names. It effectively combines classic rule-based and dictionary extractors with a contextualized language model to capture ambiguous names (e.g penny, hazel) and adapts to adversarial changes in the text by expanding its dictionary. NEAT shows 19% improvement on average in the F1 classification score for name extraction compared to previous state-of-the-art in two domain-specific datasets.

## 1 Introduction

There are approximately 4.8 million people being trafficked around the world for commercial sex, a global industry estimated to be worth $99 billion USD (Office, 2017). Technology has been a critical tool for traffickers to recruit, advertise and exploit victims (on Drugs and Crime, 2020) and the majority of human trafficking (HT) victims are advertised on online escort websites (Rhodes and Rhodes, 2016). Recently, there have been multiple efforts to leverage AI techniques for analyzing the online advertising market and providing law enforcement with actionable intelligence to counter human trafficking (Tong et al., 2017; Rabbany et al., 2018; Lee et al., 2021). One of the key tasks in this domain is *extracting information from the online escort ads* e.g. names, phone numbers, locations, prices and activity types, which are critical for higher level analysis and modus operandi detection. This information needs to be extracted from the text of the ads, and it plays a vital role in tasks such as identifying groups of related ads and finding links between them. However, this is a challenging task since in order to avoid detection, this data is made:

- *noisy* and *obscured* e.g. using 'rose' symbols as a proxy for dollar sign and spelling variations such as 'Cathy' and 'Kathy.'
- *evolving* and *adversarial* e.g. traffickers are actively introducing new patterns and intentional misspellings to avoid detection, e.g. adapting new phrases to indicate underage victims such as 'amber alert' or intentionally misspelling 'miss' as 'mizz.'

Names have a particular importance. Trafficking is an organized activity (Lee et al., 2021) and multiple people being involved in an escort ad is a strong indicator of human trafficking. Also, the more victims involved in a case, the higher priority of investigation it needs to be given in order to minimize the harm. Hence, accurately retrieving all the names in an advertisement is critical for further analysis and action.

Thus, while the general task of Named Entity Recognition (NER) usually includes diverse entities such as person names, organizations, and geolocations, in this paper, we focus on person name extraction from escort ads for combating human trafficking.

Most advanced NER models are trained on annotated structured text corpora and/or rely on contextual information for identifying entities. However, escort advertisements are usually colloquial and consist of segmented phrases instead of continuous sentences. The state-of-the-art extractor in this domain (TJBatch Nagpal et al., 2017) is rule-based

Correspondence to: pratheeksha.nair@mila.quebec



*Findings of the Association for Computational Linguistics: ACL 2022*, pages 2854 - 2868

but has limited dictionaries and has difficulty dealing with ambiguity such as distinguishing person names from location names. Therefore, simple rule-based extractors or machine learning models that are context-based alone will fail in several instances in this domain. To better illustrate this, we compared the results from TJBatch and Transformer-bert (a baseline NLP model) on two example ads collected from an escort advertisement website:

- TJBatch – The state-of-the-art named entity extractor (Dubrawski et al., 2015; Chambers et al., 2019) in the human trafficking domain. This method extracts words from a dictionary and is based on manually designed regex rules.
- Transformer-Bert – A BERT model (Devlin et al., 2018) fine-tuned on the English version of the standard CoNLL-2003 Named Entity Recognition dataset.

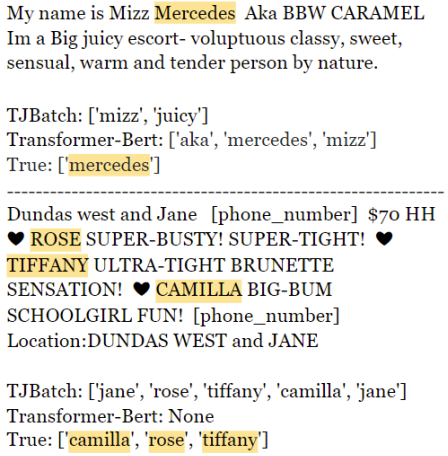



Figure 1: State-of-the-art tools fail to precisely and accurately extract all person names in an escort ads

The results (Figure 1) show that both the domain-specific and general NER models fail in identifying certain person names in the ad. In this paper, we focus on designing an improved technique for name extraction from noisy escort advertisements The main contributions of this paper are three-fold.

- We show that existing state-of-the-art tools and language models fall short in accurately identifying names from extremely noisy and unstructured escort advertisement text.
- We introduce a name extractor, NEAT (Name Extraction Against Trafficking), that enhances a core rule-based extractor with masked language models to perform disambiguation.
- We show that NEAT outperforms the previous state-of-the-art NER model for name extraction from escort advertisements, with an average of 19% improvement in F1 for our two domain-specific datasets.

## 2 Related Work

There have been several surveys and systematic reviews of the problem of named entity recognition (NER) from text (Goyal et al., 2018; Li et al., 2020; Yadav and Bethard, 2019; Saju and Shaja, 2017) which all tend to outline three broad techniques: rule-based, learning-based and hybrid approaches (Goyal et al., 2018). We group the related works as:

**Rule-based NER**: Rule-based approaches involve predefined lists, dictionaries and/or rules based on syntactic-lexical patterns which text snippets are matched against. These methods have the obvious limitation of missing out on uncommon rules and never-seen-before dictionary keys. These methods tend to fail especially in applications where text may be informal and unstructured and need to be fortified with additional modules.

**Statistical learning models for NER**: Some of the methods used for the general NER problem include Hidden Markov Models (HMM) (Wang et al., 2014), Support Vector Machines (SVM) (Saha et al., 2010), Conditional Random Fields (CRF) (Majumder et al., 2012; Wang et al., 2014), Maximum Entropy Markov Models (MEMM) (Saha et al., 2009) and Logistic Regression based systems (Ek et al., 2011). In semi-supervised approaches, a small set of the training data is first used for extracting word and context features and the rest of the data is used for training a learning algorithm in a supervised fashion (Goyal et al., 2018). However, these methods require large quantities of labelled data for training which is difficult to achieve in many real world applications, including our domain of interest.

**Embedding Models for NER**: Embedding methods such as Word2Vec (Mikolov et al., 2013a,b), GloVe (Pennington et al., 2014), and FastText (Joulin et al., 2017) generate fixed vectors for each input token and more recent methods such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) and its variations give context-aware embeddings and have been shown to improve entity extraction. A more recent work, LUKE (Yamada et al., 2020), based on a transformer model that treats not only words but also entities as independent tokens. It computes intermediate and output representations for all tokens, and reports the state-of-the-art results in general NER. Other

popular architectures for NER include bidirectional LSTM + CRF (Lample et al., 2016), enhanced with character-level contextualized representations (Akbik et al., 2018) and contextualized word representations (CWR) based on a bidirectional transformer (Alexei et al., 2019).

Applying pre-trained models directly on escort ads performs poorly because of the very limited amount of labeled data available for training. Non-context-aware models are limited in this domain due to a significant number of names that are also common nouns which may occur with an abnormal frequency due to the adversarial and pseudonymous nature of the domain (e.g. Amber, Joy, Angel...).

**NER for noisy text**: Kumar et al. (2020) conducted experiments to explore the sensitivity of BERT to synthetic noise (such as spelling mistakes) in text data and showed that performance decreases drastically with increase in noise. Mishra and Diesner (2016) introduced a linear CRF model called TwitterNER that uses random feature dropout and a modified encoding scheme in combination with semi-supervised learning from unlabelled data to handle noisy and unstructured user-generated text such as tweets. However, the low F1-score in HT datasets, as shown in Table 2, show that it is not best suited for this domain. It is worth mentioning that there are also recent efforts (Liu et al., 2021) to deal with noisy labelled NER, which is a different setting from ours.

**NER for Combating Human Trafficking**: Apart from a few, there haven't been significant efforts towards extracting names and other entities from noisy escort advertisement text. Dubrawski et al. (2015) and Nagpal et al. (2017) used rule-based approaches specifically designed for tackling this problem. This work suffers from the shortcomings of any rule-based approaches as discussed before. Kejriwal and Kapoor (2019) proposed a network-based approach that focused on the assessment of NER algorithms in the human trafficking domain that can overcome the lack of labeled evaluation data. There have also been efforts in extracting entities for general illegal activity from data scraped from Tor Darknet (Al-Nabki et al., 2020, 2019). NEAT draws on both domain-specific rule-based approaches as well embedding based models to address the shortcomings and limitations of current methods.

## 3 Problem Definition

In this section, we formally define the problem of person name extraction in the human trafficking domain both at the word-level and at the ad-level.

**Word-level definition**: Given an *advertisement* $A = (a_1, a_2, a_3, ..., a_n)$, where $a_i$ represents a sequence of words in the given ad, the NER task is to output a sequence with labels $Y = (y_1, y_2, y_3, ..., y_n)$ where $y_i \in \{0, 1\}$ ($\neg$*Person Name*, *Person Name*).

Since our focus is in correctly retrieving all the names in a given escort ad, we also consider a slightly reformulated problem:

**Ad-level definition**: Given an advertisement $A = (a_1, a_2, a_3, ..., a_n)$, we want to find a list of names that have appeared in this ad. The corresponding NER task is to output a set of words $N = \{a_i | a_i \in A \wedge y_i = 1\}$.

The language used in a typical escort ad may be both unstructured and ambiguous. The sentences are usually short and segmented using white spaces or special characters, making it harder to tokenize. It has a free choice in syntax and many person names appear in the text without a proper context. In the second example in Figure 1 the names 'Rose', 'Tiffany' and 'Camilla' are not particularly in context. Additionally, some names may be similar to location names ('Jane' in Figure 1) or adjectives (Olive, Hazel). Such names cannot simply be omitted from the dictionary either without hurting the recall. [1]

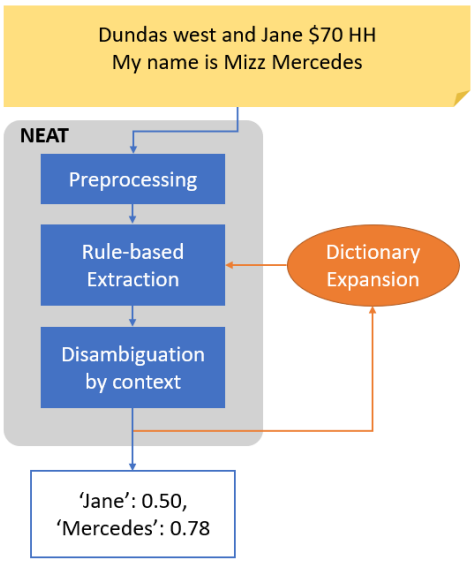
## 4 Proposed Method

We propose a person name recognition system (Figure 2) for combating human trafficking. This system consists of three modules – a rule-based extractor, a disambiguation layer and a dictionary expansion procedure. The first two modules handle named-entity recognition and disambiguation of names respectively. The third module is designed to adapt the system to evolving changes in the advertisement text like newly introduced (pseudo)names.

**Preprocessing:** Due to its colloquial nature, the input text needs to undergo preprocessing. Contractions related to a person name context are expanded (e.g "I'm" is changed to "I am") to fit the rule-based extractor, and all special characters and emojis are

[1] We define the properties of Advertisements and what we consider as a Person name in the Appendix A.

Figure 2: NEAT Overview




removed. The text is also true-cased as the extractor relies on correct part-of-speech tags.

**Rule-based extraction:** Our base extractor draws on the rule-based extraction process of TJ-Batch (Nagpal et al., 2017). It consists of 2 parts – regular expression (regex) rule matching and gazetteer matching (person name dictionary). For the regex rule matching, we manually created 15 rules, including common titles for women (e.g. 'miss') and patterns like 'my name is + NNP'. For the gazetteer matching, we collected a list of common female names as the gazetteer. The rule-based extractors find exactly matched word tokens from the input text.

*Confidence estimation:* We further measure the degree to which a pattern from the base extractors appears to be used consistently as a true person name throughout the training corpus. For a rule-based matching pattern $i$ where $i$ can be either a word in the dictionary or a regex-rule, we define its weight as:

$$p_i = \frac{cpf(i) + 1}{ctf(i) + 2}$$

where $cpf(i)$ denotes the number of times that a word is correctly predicted by the pattern $i$, and $ctf(i)$ is the number of occurrence of the pattern $i$ in the entire training corpus. Here, we are using the Laplace smoothing to account for unseen patterns.

These patterns' weights are used to assign a confidence to extracted words. More specifically, each extracted word $j$ will have two associated weights: $p_j^n$ from the pattern it is matched to with gazetteer/dictionary extractor and $p_j^r$ from the pattern it is matched to with regex-rule extractor (zero if not matched in both cases). The arithmetic mean of these two weights is considered as the total confidence of the base module on this word being a name, i.e. $p_j^b = 1/2(p_j^n + p_j^r)$.

The simple rule-based strategy, although effective, often wrongly tags certain ambiguous words as names (e.g. hazel, penny).

**Disambiguation layer:** The disambiguation layer helps to distinguish if a word extracted by the rule-based extractors is in fact a proper person name in its context. For this, we exploit masked word prediction capabilities resulting from a standard training procedure for language models such as BERT (Devlin et al., 2018). In this procedure, individual words are masked, and the model learns to predict them. Although this is typically done for training a language model on a large unlabeled corpus, we use it differently, for disambiguation.

Specifically, we use RoBERTa (Liu et al., 2019), a variant of BERT with improved pre-training, fine-tuned for the task of NER on a dataset consisting of collected escort ads (please see Section 5 for details). We call this model HT-bert.

After training, we can mask a word and get a vector of word probabilities where the highest probabilities suggest words that could reasonably replace the masked word. Therefore, if we mask a potential name, we can examine if the probable replacements are words that we are confident are names (suggesting the masked word is also a name) or that we are confident are *not* names (suggesting the masked work might not be a name) [2].

*Calibrating the confidence estimation:* Next, we re-estimate the confidence of an extracted word being a proper person name given its context based on the weight returned by the disambiguation layer, as shown in Algorithm 1. For an extracted word $j$, we calculate the weight for the disambiguation step $p_j^d$, by measuring the proportion of the predicted words in the context that are known names in our dictionary. Given the $p_j^d$ from the disambiguation step and $p_j^b$ from the base extractor, we recalibrate our confidence in $j$ being a name again as the mean of the two, i.e. $w_j = 1/2(p_j^b + p_j^d)$. Further, we set $p_j^d = -1$ if the masked word appears in the predicted words list, resulting in a final $w_j$ smaller than or equal to zero. Since it is highly unlikely

[2]More details on this model can be found in Appendix C

**Input:** a word $j$ and its context
**Require:** fill_mask function
**Require:** $k$, the number of words to predict
**Require:** $ND$, a dictionary of names
predictions = fill_mask(context of $j$);
$counter$ = 0;
**for** *each word w in predictions* **do**
  **if** *w is in ND* **then**
    $counter$++;
  **end**
**end**
$p_j^d$ = $counter$ / $k$;
**for** *each word w in predictions* **do**
  **if** *w is equal to j* **then**
    $p_j^d$ = -1
  **end**
**end**
**Return:** $p_j^d$

**Algorithm 1:** Disambiguation based on a contextual language model's fill_mask function.

for the language model to predict the exact person name based on the context. Finally, each word $j$ is accepted as a person name if $w_j$ is higher that a set threshold, which is a hyperparameter.

**Dictionary expansion:** We design a dictionary expansion module to deal with out-of-dictionary(OOD) names. Using our confidence on how likely a word is a name, $w_j$, we simply apply a different threshold/hyperparameter to add some of the more confident OOD names to our dictionary[3]; which we call **HT_filter**. We also consider combining this with two common dictionary expansion techniques: (i) **W2V** (Gentile et al., 2019) expands the dictionary by finding the neighbors of known dictionary names in the embedding space learned by a Word2Vec model, and (ii) **PFIDF** (Minkov et al., 2005) computes a Pf-Idf score that takes into consideration both the probability of a word being a name from a regex-rule extractor, and how common it is in the training corpus.

## 5 Experiment Setting

**Evaluation metrics:** We use two evaluation criteria to compare the performance of NEAT and all baseline models for person name extraction.

*Word level evaluation:* Given a list of ads $A = \{A_1, A_2, ..., A_n\}$, for an ad $A_i$, let $\hat{Y}_{A_i}$ represent the predicted set of person names and $Y_{A_i}$ represent the set of true person names. A word $w$ is defined as a True Positive (TP) instance if $w \in Y_{A_i} \cap \hat{Y}_{A_i}$, a False Negative (FN) instance if $w \in Y_{A_i} \setminus \hat{Y}_{A_i}$, and a False Positive (FP) instance if $w \in \hat{Y}_{A_i} \setminus Y_{A_i}$. Here, we report precision, recall and F1-score of word-level classification as in general NER models.

*Ad-level evaluation:* Since the proposed system is defined for a noisy text setting, we define an evaluation metric which measures the performance on a document level or an ad level. We use Intersection over Union (IoU) as the ratio of the number of common words and total number of words in the predicted set $\hat{Y}_A$ and ground truth set $Y_A$. Before calculating IoU, we split each string in $\hat{Y}_{A_i}$ and $Y_{A_i}$ by space and compute IoU based on the individual words.

$$IoU_{A_i} = \frac{Y_{A_i} \cap \hat{Y}_{A_i}}{Y_{A_i} \cup \hat{Y}_{A_i}}$$

We define the prediction of an ad $A_i$ to be a TP if $IoU_{A_i} \geq 0.5$ and a FP if $IoU_{A_i} < 0.5$. A prediction will be counted as a FN if $\hat{Y}_{A_i} = \emptyset$ and $Y_{A_i} \neq \emptyset$ and TN if $\hat{Y}_{A_i} = \emptyset$ and $Y_{A_i} = \emptyset$.
In addition to precision, recall, and F1-score, we also report F2-score for the ad-level evaluation.

$$\text{F2 score} = \frac{5 \times \text{precision} \times \text{recall}}{4 \times \text{precision} + \text{recall}}$$

This assigns more importance to recall than F1. In trafficking detection applications, it is often important to extract all the names in the text correctly (have strong recall) as this serves as the basis for downstream tasks like detecting micro-clusters of related ads (Lee et al., 2021) or finding links between connected ads (Rabbany et al., 2018).

While we acknowledge that span-level evaluation is considered standard for NER, we believe a token-level evaluation is better suited for our application. We discuss about this choice in Appendix F.

**Baselines:** We compared NEAT variants with five types of baselines, ordered by their appearance in our result tables.

*General-purpose NER packages:*
- **Stanza** (Qi et al., 2020)
- **Spacy**[4]

*BERT-based models fine-tuned for NER:*
- **Transformer-bert** (Devlin et al., 2018) – designed for general NER, trained on CoNLL2003.

[3] Details of parameter tuning is provided in Appendix D

[4] https://spacy.io/usage/linguistic-features#named-entities

| | HT1K | HT2K | CoNLL | WNUT17 |
|---|---|---|---|---|
| # of examples | 994 | 1970 | 5998 | 5690 |
| # of names | 913 | 1418 | 5623 | 2142 |
| # of unique names | 590 | 631 | 2404 | 1461 |
| # of no name examples | 267 | 885 | 3932 | 4481 |
| # of names per example | 0.92 | 0.72 | 0.94 | 0.37 |
| # of unique OOD names | 240 | 236 | 2182 | 1182 |
| # of unique in-dictionary names | 350 | 395 | 222 | 279 |
| # of non-unique in-dictionary names | 633 | 1068 | 583 | 439 |
| # of ads with more than one name | 144 | 222 | 1620 | 575 |
| dataset type | domain specific | | generic | |
| labeling method | Crowd + Manual | Crowd + Manual | Provided | Provided |

Table 1: Dataset statistics for both domain specific and generic benchmarks.

- **Fine-tuned-bert** – 'bert-base-uncased' model fine-tuned for NER.
- **Whole-mask-bert** – bert trained by masking of whole words instead of tokens, fine-tuned for NER.

*NER algorithms based on language models:*

- **Flair** (Akbik et al., 2018) – trained character language model that learns contextualized string embeddings tuned for NER.
- **ELMo** (Peters et al., 2018) – model that learns contextualized word representations tuned for NER.
- **LUKE** (Yamada et al., 2020) – transformer based model that learns contextualized embeddings, fine-tuned on TACRED dataset (Zhang et al., 2017).

*NER models for noisy text:*

- **TwitterNER** (Mishra and Diesner, 2016) – semi-supervised approach that combines text cluster information and word embeddings and uses a CRF for NER.
- **TJBatch** (Nagpal et al., 2017) – state-of-the-art rule-based named entity extractor in the human trafficking domain.

*RoBERTa with and without domain adaptation:*

- **Fine-tuned-roberta** (Liu et al., 2019) – RoBERTa language model with improved pre-training, fine-tuned for NER.
- **HT-bert** – RoBERTa model further trained on unsupervised masked word prediction with 1.1 million escort ads[5], then fine-tuned for NER.

NEAT *variants:*

- **NEAT-base** - rule-based extractors only that returns all the matched words regardless of the confidence threshold.
- **NEAT-fixed** - NEAT model including disambiguation layer and confidence threshold. The dictionary used is fixed across all datasets.
- **NEAT-update** - full NEAT model with augmented dictionary using true names from the training set. The weights of newly added words are computed by the same method used to compute $p_i$.
- **NEAT** - full NEAT model with augmented dictionary using one pass of dictionary expansion module.

**Datasets:** The performance of NEAT and all baseline models was evaluated on four datasets – two new escort ad datasets we curated ourselves and two general NER datasets common in the literature (CoNLL 2003 Sang and Meulder, 2003 and WNUT17 Derczynski et al., 2017). We split all datasets 80-20 train-test. For hyperparameter tuning, we split another 20% from the train set for validation. The summary of the datasets can be found in Table 1. In particular, we have:

- **HT1K:** 994 examples crawled from an escort advertisement website.[5] Names were annotated by crowdsourcing through Amazon Mechanical Turk, plus a member of our team resolving conflicting cases. 75.6% of the examples were labeled with the majority of crowd workers in agreement, 12.2% were resolved manually, and the remaining 12.2% were excluded from the test set. More details on the annotation process can be found in Appendix B.

  For training HT-bert, a separate batch of 1.1 million escort ads were also crawled from this source.
- **HT2K:** 1970 examples crawled from a different escort advertisement website,[6] with annotations similar to the HT1K dataset. In this case 88.7%

[5]We omit the website name here to avoid any potential impacts on the website or on individuals in our data

of the labels use the crowd worker majority, 6.8% were resolved manually, and the last 4.5% were excluded from the test set.

- **CoNLL2003**: 6000 examples. The general dataset includes many types of entities; we evaluate both this general prediction (for models that are not designed specifically for names) and the names only (the B-PER and I-PER annotations).
- **WNUT17:** 5689 examples. We restrict evaluation to the names (B-PER and I-PER).

## 6 Results and Discussion

**Name Extraction:** We evaluate the performance of the baseline models and NEAT for name extraction from text and display the results in tables 2, 3 and 4. We see that NEAT gives the best overall F1 score performance in the word-level evaluation (Table 2) on both domain-specific datasets: HT1K and HT2K. In Table 3, we perform equally well in ad-level evaluation on HT1K and HT2K in both F1 and F2 metrics.

It's worth mentioning that NEAT has better performance on HT2K than on HT1K. This outcome is not surprising because our weight assignments are trained with HT2K training set explicitly. Since advertisements from different websites have different language features and it is unrealistic for our tool to be retrained on every new website we crawl, it's important for our proposed system to do effective transfer learning. Results in Table 2 and Table 3 support that even though the assignments are trained on HT2K, NEAT still performs relatively well in F1 (and F2), outperforming all other models.

Conversely, both NEAT and TJBatch perform worse on the non-HT datasets. This is likely because their rule-based components are specifically designed to capture patterns common in escort ads but less so in generic data. We found that using our regex-rule extractor alone, without the rest of our model, NEAT would only extract 3 words from all of CoNLL and 10 from all of WNUT17, confirming the rules do not fit these datasets. The dictionary component of our model also lacks many traditional male names and last names of all genders.

We include an ablation study to review the effectiveness of modules. Results from **NEAT-base** and **NEAT-fixed** performs well on the domain datasets but poorly on the generic datasets due to the reason mentioned above. **NEAT-fixed** in general performs better than **NEAT-base** because of the additional disambiguation module.

Finally, we see that HT-bert, as well as the other BERT-based models, gives mediocre performance in this domain. This shows that standard methods for domain adaptation of language models (i.e. unsupervised training on a domain-specific corpus, and supervised fine-tuning) are insufficient to produce optimal results. An approach that can leverage stronger domain adaptation, like ours, is needed.

**Parameter tuning:** We investigate the consequence of adding weight components and experiment with different parameter combinations (of threshold and $k$) on the HT2K dataset. The threshold is the minimum score a word extracted by NEAT needs to have to be considered a candidate in the list of words predicted by HT-bert 'fill_mask'. The dataset is split into train, validation and test sets. $M_{base}$ represents the performance measures of the base extractors alone without weights. $M_{weighted}$ denotes the performance measures of the proposed weighted extractor. The results in Table 5 are relative changes in the metrics calculated as $(M_{weighted} - M_{base})/M_{base}$. The table demonstrates that adding weight components can effectively disambiguate the output words thereby increasing precision. The F1-score increase reaches its peak when the weights threshold is set to be 0.1. On the other hand, when the parameter k of fill_mask equals 40, the relative metrics have the best overall scores. The results in the preceding section use this best parameter pair.

**Dictionary Expansion:** We study the effect of dictionary expansion and experiment with different methods on the HT2K dataset as shown in Table 6. We first randomly remove half of the names in the dictionary and run the dictionary expansion using three methods: W2V, PFIDF, and NEAT. The threshold value is tuned on the validation set for each method (tuning results for W2V and PFIDF are shown in Appendix D). $M_{half\_dict}$ represents the performance measures of the dictionary extractor using only half of the original dictionary. $M_{expanded}$ denotes the performance measures of dictionary extractor after dictionary expansion. The results in Table 6 are relative metrics calculated by $(M_{expanded} - M_{half\_dict})/M_{half\_dict}$ and 'count' indicates the number of new words added to the dictionary after expansion. The goal for dictionary expansion is to tackle the unseen word problem in a gazetteer based extractor and increase the recall rate. From the experiment results, recall is greatly

| Method | HT1K | | | HT2K | | | CoNLL2003 | | | WNUT17 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | Prec | Rec | F1 | Prec | Rec | F1 | Prec | Rec | F1 | Prec | Rec |
| Stanza | .41±.01 | .40±.01 | .41±.01 | .38±.04 | .43±.04 | .35±.05 | .83±.01 | .83±.01 | .83±.02 | .64±.03 | .66±.03 | .62±.03 |
| Spacy | .22±.03 | .19±.03 | .27±.03 | .19±.02 | .18±.02 | .21±.03 | .74±.01 | .81±.01 | .69±.01 | .44±.01 | .46±.02 | .42±.01 |
| Transformer-bert | .20±.04 | .57±.11 | .12±.02 | .46±.06 | .63±.07 | .36±.04 | .84±.01 | .85±.01 | .83±.01 | .55±.01 | .67±.01 | .47±.02 |
| Fine-tuned-bert | .56±.02 | .69±.01 | .47±.02 | .60±.01 | .76±.03 | .49±.01 | .96±.01 | .98±.00 | .95±.01 | **.78**±.02 | **.83**±.01 | .73±.02 |
| Whole-mask-bert | .31±.01 | .45±.02 | .25±.01 | .41±.03 | .38±.04 | .56±.01 | .87±.01 | .86±.01 | .86±.00 | .42±.01 | .69±.00 | .30±.03 |
| Flair | .40±.04 | .65±.06 | .29±.03 | .37±.03 | .61±.03 | .27±.02 | **.98**±.00 | **.99**±.00 | .96±.01 | .62±.01 | .65±.02 | .59±.02 |
| ELMO | .49±.03 | .56±.06 | .43±.02 | .46±.05 | .59±.05 | .38±.04 | .97±.01 | .97±.01 | **.98**±.01 | .64±.02 | .67±.02 | .62±.03 |
| Luke | .50±.03 | .80±.03 | .36±.02 | .49±.02 | .73±.04 | .37±.01 | .28±.02 | .94±.01 | .16±.01 | .45±.01 | .55±.01 | .38±.02 |
| TwitterNER | .52±.04 | **.81**±.03 | .38±.04 | .47±.03 | .71±.06 | .35±.02 | .41±.01 | .90±.01 | .26±.01 | .57±.01 | .81±.01 | .44±.01 |
| TJBatch | .67±.02 | .64±.01 | .71±.04 | .64±.02 | .63±.03 | .65±.02 | .10±.00 | .40±.03 | .06±.00 | .18±.01 | .31±.01 | .13±.01 |
| Fine-tuned-roberta | .57±.02 | .69±.01 | .49±.03 | .59±.01 | .76±.04 | .59±.03 | .96±.01 | .98±.00 | .94±.01 | **.78**±.05 | .81±.03 | **.72**±.04 |
| HT-bert | .58±.02 | .71±.03 | .49±.02 | .60±.02 | .76±.04 | .50±.02 | .95±.01 | .96±.01 | .93±.01 | .73±.01 | .79±.04 | .67±.02 |
| NEAT-base | .74±.01 | .77±.02 | .72±.03 | .73±.02 | .69±.03 | .77±.03 | .17±.01 | .50±.02 | .10±.01 | .29±.01 | .51±.03 | .21±.01 |
| NEAT-fixed | .75±.01 | .81±.03 | .70±.03 | .79±.03 | **.80**±.03 | .77±.03 | .17±.01 | .50±.02 | .10±.01 | .26±.02 | .48±.03 | .18±.01 |
| **NEAT-update** | **.76**±.01 | .80±.02 | **.73**±.03 | **.81**±.02 | .79±.02 | **.83**±.03 | .69±.01 | .82±.01 | .59±.01 | .47±.01 | .55±.02 | .41±.02 |
| **NEAT** | **.76**±.01 | **.81**±.02 | .71±.03 | .80±.02 | .79±.03 | .79±.03 | .17±.01 | .50±.02 | .10±.01 | .27±.02 | .48±.03 | .19±.01 |

Table 2: Word-level strict match results on 5-split test sets: F1 score, Precision, Recall. NEAT gives a significant improvement on HT domain datasets (HT1K and HT2K).

| Method | HT1K | | | | HT2K | | | |
|---|---|---|---|---|---|---|---|---|
| | F1 | F2 | Prec | Rec | F1 | F2 | Prec | Rec |
| Stanza | .47±.01 | .47±.01 | .47±.02 | .48±.02 | .47±.05 | .44±.05 | .51±.05 | .43±.05 |
| Spacy | .30±.03 | .31±.04 | .25±.03 | .33±.04 | .26±.02 | .27±.02 | .24±.02 | .29±.02 |
| Transformer-bert | .22±.03 | .17±.04 | .52±.07 | .14±.02 | .56±.04 | .46±.05 | .63±.07 | .36±.04 |
| Fine-tuned-bert | .65±.01 | .64±.01 | .67±.03 | .63±.02 | .65±.02 | .58±.03 | .82±.01 | .54±.03 |
| Whole-mask-bert | .41±.02 | .39±.01 | .46±.03 | .38±.00 | .55±.02 | .64±.01 | .44±.00 | .79±.02 |
| Flair | .45±.03 | .35±.03 | .77±.07 | .31±.02 | .42±.02 | .33±.01 | .73±.04 | .29±.01 |
| ELMO | .56±.04 | .51±.03 | .67±.06 | .49±.02 | .55±.04 | .49±.03 | .67±.04 | .46±.04 |
| Luke | .58±.03 | .49±.03 | .83±.03 | .45±.04 | .54±.04 | .46±.03 | .76±.06 | .42±.03 |
| TwitterNER | .60±.04 | .52±.05 | .80±.03 | .48±.05 | .53±.03 | .45±.03 | .76±.06 | .41±.03 |
| TJBatch | .84±.02 | .87±.02 | .78±.03 | **.89**±.03 | .76±.03 | .77±.02 | .74±.03 | .78±.02 |
| Fine-tuned-roberta | .65±.01 | .64±.01 | .68±.01 | .64±.01 | .65±.03 | .57±.03 | .82±.03 | .53±.04 |
| HT-bert | .66±.01 | .64±.01 | .69±.03 | .64±.01 | .66±.02 | .59±.03 | .83±.03 | .55±.03 |
| NEAT-base | **.87**±.02 | .87±.02 | .86±.03 | .88±.02 | .82±.02 | .86±.01 | .76±.04 | .90±.02 |
| NEAT-fixed | .86±.02 | .85±.02 | **.88**±.03 | .85±.02 | .87±.03 | .89±.03 | .84±.04 | .90±.03 |
| **NEAT-update** | **.87**±.02 | .88±.01 | .86±.02 | .88±.02 | **.88**±.02 | **.90**±.02 | **.85**±.03 | .91±.02 |
| **NEAT** | **.87**±.01 | **.88**±.02 | .85±.03 | **.89**±.02 | **.88**±.02 | **.90**±.02 | .83±.04 | **.93**±.02 |

Table 3: Ad-level match results: F2 score, Precision, Recall. As in the previous table, NEAT gives a significant improvement on HT domain datasets (HT1K and HT2K).

increased as expected. It is also worth noticing the NEAT_filter method has a slight increase in precision after the expansion. This result shows that most of the expanded words from NEAT_filter are actual person names in their context. NEAT_filter also has the highest F1 increase among the three primary methods. However, PFIDF finds more words than NEAT_filter while having a comparable F1 increase. By combining both, we get the highest increase in F1 and recall.

While this seems effective for improving this halved version of our dictionary, we found it did not significantly improve the performance of our overall model with the full dictionary as the result of **NEAT** suggests. We hypothesize this is because the full dictionary is already saturated for the domain specific datasets and hard to improve. Also, since regex-rule can only match a few instances in the generic corpus, the corresponding candidate pool for expanding the dictionary is limited. Nonetheless, these results show this procedure can be helpful as obscuration and adversarial strategies evolve and the dictionary needs improvement.

An additional observation is that after we included all the true names from the training sets and rerun NEAT, the performance of **NEAT-update** shows significant performance increases in the generic datasets. This suggests that we can adapt our system to other topic specific corpus by choosing the appropriate dictionary.

| Method | CoNLL2003 | | | | WNUT17 | | | |
|---|---|---|---|---|---|---|---|---|
| | F1 | F2 | Prec | Rec | F1 | F2 | Prec | Rec |
| Stanza | .84±.01 | .84±.01 | .83±.01 | .85±.01 | .66±.03 | .66±.02 | .66±.04 | .66±.02 |
| Spacy | .75±.01 | .74±.01 | .77±.02 | .74±.01 | .44±.02 | .44±.02 | .44±.02 | .44±.01 |
| Transformer-bert | .88±.01 | .84±.01 | .85±.01 | .83±.01 | .58±.01 | .55±.01 | .67±.01 | .47±.02 |
| Fine-tuned-bert | .97±.01 | .97±.01 | .97±.01 | .97±.01 | **.79**±.02 | **.79**±.03 | **.79**±.02 | **.79**±.04 |
| Whole-mask-bert | .83±.02 | .84±.01 | .85±.02 | .82±.03 | .49±.01 | .43±.00 | .63±.01 | .40±.02 |
| Flair | **.98**±.00 | **.98**±.01 | **.99**±.00 | **.98**±.01 | .62±.02 | .65±.02 | .59±.02 | .63±.05 |
| ELMO | **.98**±.00 | **.98**±.00 | .97±.00 | **.98**±.01 | .64±.02 | .65±.03 | .64±.02 | .65±.04 |
| Luke | .40±.03 | .32±.02 | .69±.04 | .29±.02 | .52±.01 | .53±.02 | .5±.01 | .54±.02 |
| TwitterNER | .55±.01 | .50±.01 | .64±.02 | .48±.02 | .67±.01 | .65±.01 | .71±.01 | .64±.02 |
| TJBatch | .14±.00 | .11±.00 | .24±.02 | .10±.00 | .19±.01 | .17±.01 | .21±.01 | .17±.01 |
| Fine-tuned-roberta | .97±.01 | .97±.01 | .97±.01 | .97±.01 | .77±.03 | .78±.04 | .77±.03 | .78±.04 |
| HT-bert | .95±.01 | .96±.01 | .95±.02 | .96±.01 | .74±.01 | .74±.01 | .75±.04 | .74±.03 |
| NEAT-base | .21±.01 | .19±.01 | .28±.01 | .18±.01 | .32±.02 | .28±.01 | .40±.03 | .26±.01 |
| NEAT-fixed | .21±.01 | .18±.02 | .28±.01 | .17±.02 | .28±.02 | .24±.02 | .36±.03 | .23±.02 |
| **NEAT-update** | .74±.02 | .77±.02 | .71±.02 | .78±.02 | .51±.01 | .54±.03 | .48±.01 | .55±.04 |
| **NEAT** | .21±.01 | .18±.01 | .28±.02 | .17±.01 | .29±.02 | .25±.02 | .37±.03 | .24±.02 |

Table 4: Ad-level match results: F2 score, F1 score, Precision, Recall on CoNLL2003 and WNUT17.

| Thres | k | | | | | |
|---|---|---|---|---|---|---|
| | 30 | | **40** | | 50 | |
| | F1 | Prec | F1 | Prec | F1 | Prec |
| 0.08 | +4.0 | +11.1 | +4.4 | +12.0 | +4.4 | +12.0 |
| 0.09 | +4.3 | +17.2 | +4.3 | +17.3 | +4.3 | +16.6 |
| **0.10** | +4.8 | +20.1 | +4.8 | +20.1 | +4.8 | +20.1 |
| 0.11 | +4.1 | +21.6 | +4.1 | +21.6 | +4.1 | +21.6 |
| 0.12 | +3.7 | +22.4 | +3.8 | +22.5 | +3.8 | +22.4 |
| 0.13 | -0.7 | +26.1 | -0.3 | +25.1 | -0.3 | +25.1 |

Table 5: Percentage change in word-level performance on HT2K test set using different parameters for the weighted extractor and disambiguation.

| Method | F1 | Pre | Rec | count |
|---|---|---|---|---|
| W2V | +1.9% | -8.4% | +10.3% | 52 |
| PF-IDF | +9.8% | -1.2% | +18.9% | 82 |
| NEAT_filter | +10.9% | +2.8% | +17.2% | 48 |
| Combined | +13.9% | +0.1% | +25.9% | 98 |

Table 6: Percentage change in performance scores using different dictionary expansion methods on HT2K test set.

## 7 Conclusions

We presented NEAT, which addresses the problem of name extraction from noisy, adversarial escort advertisement text. NEAT consists of two main components - a rule-based extractor (combining a dictionary of names and regex matching) and a disambiguation layer.

NEAT significantly improves on the previous state-of-the-art for this task, with around 19 percentage points at the word-level and 9 percentage points at the ad-level improvements in F1 score on two datasets. Both our method and the previous state-of-the-art outperform generic methods, highlighting the continued need for and benefit of domain-driven approaches.

By modifying the dictionary and regex matching, our pipeline can be adapted to other domains and tasks to a greater extent than is possible with many generic models and methods. In future work, we plan to investigate this further, by examining other tasks (such as extracting locations as well as person names) and domains (such as more generic but still noisy, Twitter data).

**Reproducibility:** Our code is made available online at https://github.com/tudou0002/NEAT. The domain related data used in this paper contains person identifying information and can be made available for research purposes only, based on a data-sharing agreement. Please reach out to the authors of the paper for getting access to this data.

## Acknowledgements

This research is partially funded by the Canada CIFAR AI Chairs Program. It is also partially supported by Samsung-Mila Research Grant on Entity Extraction from Noisy Adversarial Data. The third author receives funding from IVADO.

## References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

Mhd Wesam Al-Nabki, Eduardo Fidalgo, and Javier Velasco Mata. 2019. Darkner: A platform for named entity recognition in tor darknet. *Jornadas Nacionales de Investigación en Ciberseguridad (JNIC2019)*, 1:279–280.

Mhd Wesam Al-Nabki, Francisco Jañez-Martino, Roberto A Vasco-Carofilis, Eduardo Fidalgo, and Javier Velasco-Mata. 2020. Improving named entity recognition in tor darknet with local distance neighbor feature. *arXiv preprint arXiv:2005.08746*.

Baevski Alexei, Edunov Sergey, Liu Yinhan, Zettlemoyer Luke, and Auli Michael. 2019. Cloze-driven pretraining of self-attention networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China.

Nathanael Chambers, Timothy Forman, Catherine Griswold, Kevin Lu, Yogaish Khastgir, and Stephen Steckler. 2019. Character-based models for adversarial phone extraction: Preventing human sex trafficking. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 48–56, Hong Kong, China. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Artur Dubrawski, Kyle Miller, Matthew Barnes, Benedikt Boecking, and Emily Kennedy. 2015. Leveraging publicly available data to discern patterns of human-trafficking activity. *Journal of Human Trafficking*, 1(1):65–85.

Tobias Ek, Camilla Kirkegaard, Håkan Jonsson, and Pierre Nugues. 2011. Named entity recognition for short text messages. *Procedia-Social and Behavioral Sciences*, 27:178–187.

A. L. Gentile, D. Gruhl, P. Ristoski, S. Welch, and Eswc th th International Semantic Web Conference. 2019. Explore and exploit. dictionary expansion with human-in-the-loop. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11503 LNCS:131–145.

Archana Goyal, Vishal Gupta, and Manish Kumar. 2018. Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, 29:21–43.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431. Association for Computational Linguistics.

Mayank Kejriwal and Rahul Kapoor. 2019. Network-theoretic information extraction quality assessment in the human trafficking domain. *Applied Network Science*, 4(1):44.

Ankit Kumar, Piyush Makhija, and Anuj Gupta. 2020. Noisy text data: Achilles' heel of BERT. In *Proceedings of the Sixth Workshop on Noisy User-generated Text, W-NUT@EMNLP 2020 Online, November 19, 2020*, pages 16–21. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics.

Meng-Chieh Lee, Catalina Vajiac, Aayushi Kulshrestha, Sacha Levy, Namyong Park, Cara Jones, Reihaneh Rabbany, and Christos Faloutsos. 2021. Infoshield: Generalizable information-theoretic human-trafficking detection. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1116–1127. IEEE.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon

Shashua, and Yoav Shoham. 2020. Sensebert: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4656–4667. Association for Computational Linguistics.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Kun Liu, Yao Fu, Chuanqi Tan, Mosha Chen, Ningyu Zhang, Songfang Huang, and Sheng Gao. 2021. Noisy-labeled NER with confidence estimation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3437–3445. Association for Computational Linguistics.

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

Mukta Majumder, Utsav Barman, Rahul Prasad, Kumar Saurabh, and Sujan Kumar Saha. 2012. A novel technique for name identification from homeopathy diagnosis discussion forum. *Procedia Technology*, 6:379–386.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 443–450, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Shubhanshu Mishra and Jana Diesner. 2016. Semi-supervised named entity recognition in noisy-text. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 203–212.

Chirag Nagpal, Kyle Miller, Benedikt Boecking, and Artur Dubrawski. 2017. An entity resolution approach to isolate instances of human trafficking online. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 77–84. Association for Computational Linguistics.

International Labour Office. 2017. Global estimates of modern slavery.

United Nations Office on Drugs and Crime. 2020. Global report on trafficking in persons.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. 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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 157–163. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Reihaneh Rabbany, David Bayani, and Artur Dubrawski. 2018. Active search of connections for case building and combating human trafficking. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2120–2129.

Leanne Maree Rhodes and Leanne Maree Rhodes. 2016. Human trafficking as cybercrime. *Agora Int J Admn Sci*, 1(1):23–29.

Sujan Kumar Saha, Shashi Narayan, Sudeshna Sarkar, and Pabitra Mitra. 2010. A composite kernel for named entity recognition. *Pattern Recognition Letters*, 31(12):1591–1597.

Sujan Kumar Saha, Sudeshna Sarkar, and Pabitra Mitra. 2009. Feature selection techniques for maximum entropy based biomedical named entity recognition. *Journal of biomedical informatics*, 42(5):905–911.

C Janarish Saju and AS Shaja. 2017. A survey on efficient extraction of named entities from new domains using big data analytics. In *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*, pages 170–175. IEEE.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating human trafficking with multimodal deep models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1547–1556.

Yaqiang Wang, Zhonghua Yu, Li Chen, Yunhui Chen, Yiguang Liu, Xiaoguang Hu, and Yongguang Jiang. 2014. Supervised methods for symptom name recognition in free-text clinical records of traditional chinese medicine: an empirical study. *Journal of biomedical informatics*, 47:91–104.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *CoRR*, abs/1910.11470.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6442–6454. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

## A Appendix: Terminology

Here we define key terminology used in the scope of this work and briefly discuss some of its properties.

- *Advertisement*. In the scope of this paper, an online *advertisement* is defined as a notice or announcement in a public website promoting a sexual service.

  *Unstructured Text.* The sentences in online escort ads are usually short and segmented using white spaces or special characters, making it harder to split the sentences correctly. Also, since online advertisement has a relatively free choice in syntax, many person names appear in the text without a proper context. Since we cannot accurately predict the next word based on the context, a simple language model cannot play to its strength in these scenarios. In the second example in Figure 1 the names 'Rose', 'Tiffany' and 'Camilla' are not particularly in context.

  *Ambiguity.* Another significant challenge is the ambiguity in the text. Person names can be very similar to some of the location names ('Jane' in 1). Additionally, certain names may also be adjectives (Penny, Hazel), thus confusing dictionary and rule-based extractors. Removing these ambiguous words from the dictionary can partially solve this, but then we face a decrease in the recall rate.
- *Person Name*. A *Person Name* that appears in an advertisement is usually associated with other pronouns or adjectives. We declare a word as a person name if it is both a person name in a regular context and a word that refers to the person mentioned in the advertisement. Some names may also have varying spellings or be misspelled.

## B Appendix: Annotated Process

As noted in the main paper, to get the base for HT1K, we randomly selected 1000 examples from a large dataset of escort advertisements crawled from a single website. We followed the same procedure for HT2K, but with 2000 examples and from another website. We then crowdsourced annotations with Amazon Mechanical Turk. Examples were preprocessed to remove special characters, such as emoji, which could not be input into Mechanical Turk. Each example was annotated by 3 different workers using IE-Turk,[6] a package that gives workers an interface where they can click on or highlight named entities to mark them. Workers were required to be Mechanical Turk Masters (a designation Amazon awards for consistent quality) and to have at least an 85% approval rate on all tasks. Amazon also requires workers to have Adult Content Qualification for a task like this with sensitive content. Workers were paid $0.03 per example, which lead to fast completion.

The short-form instructions were "Select (or click) all person names in the text." The long-form instructions were:

> Select all words in the text that name a person, including repeats. Some names may contain spaces, other characters, or have other issues with the writing. Please ignore those issues and

[6] https://github.com/Varal7/ieturk

select the name anyways. For example, if the text contains "Alex ander" or "Alex@nder" or "HiImAlexander" or "Alexander@email.com," please select all of them. In the latter two cases, it will only let you select the entire "word," but that is fine. Pronouns like she/her/he/his do not count as names. Not all text will contain names; if there is no name, please check the "There is no person name" box and continue.

Your annotations will be used for research to help fight human trafficking.

IE-Turk does not differentiate between two successive names and a first name plus last name. Therefore, similar to other NER datasets like CoNLL and WNUT17, our labels are individual words.

The annotator agreement is given in Table 7. There are four cases. When none of the annotators agree, we take one randomly for training examples, and do not use them for evaluation. When two of the annotators find the same names, or all three annotators agree, we follow the standard procedure of taking the majority vote.

The most complex case is when two of the annotators agree on an empty set (i.e. find no names) but the third does find one or more names. The conventional approach might be to still take the majority vote, but we observed many valid names among the minority annotation. Therefore, one of our team members manually resolved the labels for these cases to produce a more accurate result. Based on these stronger labels, these cases are likely to contain a name: 82.6% of such cases in HT1K, and 56.1% in HT2K, contained at least one name. This suggests that if additional annotation is not possible, it will be optimal to use the label from the single annotator who found one or more names, rather than the majority who found no names. However, for best quality, it is necessary to either drop or re-annotate these cases, as we did for these datasets.

A small number of examples which produced errors or ambiguity during the annotation or its post-processing were removed, leaving 994 and 1970 examples in HT1K and HT2K respectively.

The "Two (non-empty)" and "All" categories, which reflect the strongest agreement from the crowd workers, comprise most of the examples – over 75% in HT1K and over 88% in HT2K. Thus, this crowdsourcing is effective in producing accurate labeled examples for this domain. In combination with our manual resolutions for the "Two (empty)" category, we obtain accurate labels for over 85% and 95% of the original data.

| Workers in Agreement | HT1K | HT2K |
|---|---|---|
| None | 121 (12.2%) | 88 (4.5%) |
| Two (empty) | 121 (12.2%) | 134 (7.0 %) |
| Two (non-empty) | 302 (30.4%) | 399 (20.3%) |
| All | 450 (45.2%) | 1349 (68.5%) |

Table 7: Number of examples grouped by worker agreement. Two or more workers agree on the majority of examples.

## C Appendix: Disambiguation with Masked Word Prediction

We exploit masked word prediction capabilities resulting from a standard training procedure for language models such as BERT (Devlin et al., 2018). In this procedure, individual words are masked, and the model learns to predict them. This is typically done using weight tying (Levine et al., 2020; Inan et al., 2017; Press and Wolf, 2017). First, one-hot vectors representing each token, including the special mask token, are converted to input embedding vectors through multiplication by a matrix $W$. After adding a positional encoding, these embeddings are passed through the core transformer part of the model. This produces output embeddings for each token. In weight tying, the output embedding of the mask token is then decoded to word predictions using $W^T$, the transpose of the matrix that encodes the inputs. Finally, after applying a softmax, the predictions are trained with a cross-entropy loss to approach the one-hot coding of the real token that was masked.

Typically, this procedure provides an effective way of training a language model on a large unlabeled corpus. We use it differently, for disambiguation. For each word extracted by the rule-extractor as a potential name, we mask it in its sentence and use HT-bert 'fill_mask' to obtain a confidence score for that word being a valid person name.

## D Appendix: Parameter Tuning for other Dictionary Expansion Methods

Table 8 shows the parameter tuning for dictionary expansion using Word2Vec, Table 9 shows results for different thresholds using PF-IDF and Table 10 shows results with different thresholds on weights

| Top_k \ Thres | 2 | 5 | 8 |
|---|---|---|---|
| **5** | +1.0% | -3.0% | +7.0% |
| 10 | -10.8% | -3.1% | +4.3% |
| 15 | -14.9% | +0.1% | +6.1% |

Table 8: Parameter tuning for W2V. The best/bold setting is used for Table 6. Change in F1-score using different parameters for the Word2Vec method.

| Threshold | F1 | Precision | Recall |
|---|---|---|---|
| **0.1** | +13.8% | -0.6% | +26.0% |
| 0.3 | +3.1% | -3.2% | +8.0% |
| 0.5 | +1.6% | -1.7% | +4.0% |

Table 9: Parameter tuning for PF-IDF. The best/bold setting is used for Table 6. The table shows percentage change in metrics using different threshold for Pfidf.

in NEAT.

## E Appendix: Baseline models

*General-purpose NER packages:*

- **Stanza** (Qi et al., 2020) is a collection of accurate and efficient tools for the linguistic analysis developed by Stanford NLP group. The named entity recognition (NER) module in Stanza recognizes mention spans of a particular entity type in the input sentence.
- **Spacy** is an open-source python-based software library for NLP tasks including NER.

*BERT-based models fine-tuned for NER:*

- **Transformer-bert** (Devlin et al., 2018) uses the BERT model from the NER pipeline as implemented by Huggingface
- **Fine-tuned-bert** (Devlin et al., 2018) is a 'bert-base-uncased' model that has been fine-tuned for NER using a combination of CoNLL2003 and names extracted from escort advertisements.
- **Whole-mask-bert** (Devlin et al., 2018) uses a BERT model that has been trained by masking entire words at random instead of tokens. We use the 'bert-large-cased-whole-word-masking' model and fine-tune it for the task of NER using the same combined dataset as the other baselines.

| Threshold | F1 | Precision | Recall |
|---|---|---|---|
| 0.10 | +10.1% | -3.7% | +21.9% |
| **0.12** | +11.2% | +2.5% | +28.0% |
| 0.14 | +9.8% | +1.8% | +16.0% |

Table 10: Parameter tuning for dictionary expansion using NEAT. The overall best/bold setting is used for Table 6. Change in metrics using different threshold on weights in NEAT.

*NER algorithms based on language models:*

- **ELMo** (Peters et al., 2018) is a context word representation based on deep bidirectional LSTM used for the downstream task of NER.
- **LUKE** (Yamada et al., 2020) is a transformer based model that treats words and entities as independent tokens and learns contextualized representations. The LUKE model fine-tuned on TACRED dataset (Zhang et al., 2017) as implemented in the Hugging Face library is used for the experiments.
- **Flair** (Akbik et al., 2018) uses the internal states of a trained character language model to produce a novel type of word embedding known as as contextual string embeddings which is used for the downstream task of NER.

*NER models for noisy text:*

- **TwitterNER** (Mishra and Diesner, 2016) designed for noisy and unstructured text such as tweets uses a semi-supervised approach which includes text cluster information along with word embeddings and employs random feature dropout and a CRF for entity recognition.
- **TJBatch** (Nagpal et al., 2017) The state-of-the-art named entity extractor in the human trafficking domain. It is able to extract more than 20 kinds of entity types using the rule-based strategy. We only consider the person name entity type in our experiment.

*RoBERTa with and without domain adaptation:*

- **Roberta-base** (Liu et al., 2019) is a language model based on BERT, with modifications to the pre-training to improve efficiency and performance. We fine-tune it for NER in the same way as fine-tuned-bert.
- **HT-bert** adapts Roberta-base to the HT domain by training unsupervised masked word prediction on 1.1 million ListCrawler escort ads. It is then fine-tuned for name extraction, again as with fine-tuned-bert.

As a pre-processing step, all emojis and non-ASCII characters are removed from the advertisement texts before passing them through the NER models. Each dataset was split into 80-20 train-test split and the train set was used to fine-tune the BERT-based models. We do-not distinguish

between 'B-PER' and 'I-PER' in the labels (i.e if they appear in the beginning or in the middle of the text). For evaluation, both the predicted and ground truth names are converted to lower case.

## F Appendix: Token-level Evaluation Metric

Due to the informal language nature of the escort advertisements, most, if not all, of the person name in the descriptions are first name or nickname basis. In these cases, the token-level evaluation and the span-level evaluation will end up having the same performance. For example, "*Lia available now*" with "*Lia*" predicted as **PER** has the same precision and recall score irrespective of whether the true label sequence is [B-PER, O, O] or [PER, O, O].

Moreover, the token-level evaluation can be considered as a partial span-level evaluation where the constraints on the entity boundaries are loosened. We still gain valid information about a person being advertised even if the prediction model gets the entity boundary wrong. For example, using our token-level evaluation, "*Jane Smith left*" with a true label [B-PER, I-PER, O] will have the same performance metric whether the predicted sequence is [B-PER, O, O] or [O, B-PER, O].

Lastly, in the context of name extraction from escort ads, it is common to see the same name appearing multiple times consecutively in a single ad. This could be done in order to make a better impression of the persons being advertised. The span-level sequence evaluation may count every correctly predicted occurrence of a name, regardless of whether it has already been extracted. This may give extra false credibility to the extraction algorithm since what we really want to measure is the individual being advertised.