Attribute Extraction from Noisy Text Using Character-based Sequence Tagging Models

Pallika Kanani
Oracle Labs
pallika.kanani@orcle.com

Michael Wick
Oracle Labs
michael.wick@oracle.com

Adam Pocock
Oracle Labs
adam.pocock@oracle.com

Abstract

Attribute extraction is the problem of extracting structured key-value pairs from unstructured data. Many similar entity recognition problems are usually solved as a sequence labeling task in which elements of the sequence are word tokens. While word tokens are suitable for newswire, for many types of data—from social media text to product descriptions–word tokens are problematic because simple regular-expression based word tokenizers can not accurately tokenize text that is inconsistently spaced. Instead, we propose a character-based sequence tagging approach that jointly tokenizes and tags tokens. We find that the character-based approach is surprisingly accurate both at tokenizing words, and at inferring labels. We also propose an end-to-end system that uses pairwise entity linking models for normalizing the extracted values.

1 Introduction

A large amount of text found in commercial databases is unstructured and noisy. Often, there are multiple pieces of information contained in this text, which would be much more valuable if available in structured form, such as key value pairs. These values can then form the basis of advanced analytics and predictive models. The process of extracting key-value pairs from text is known as *attribute extraction*, Ghani et al. [2, 10]. Attributes are more useful than raw text for application areas like e-commerce and retail analytics. In many industrial settings, significant manual effort is employed in cleaning up this text, and extracting structured key value pairs from it.

An important part of automating this process is to cast it as a sequence labeling problem with labels as keys and tokens as values. In the sequence labeling problem, the observed elements of the sequence are tokens, each of which is associated with a hidden label variable. Thus, tokenization is a prerequisite that we take for granted as a solved problem. Indeed, for many types of named entity recognition tasks, such as for newswire text, tokenization is easily solved with simple regular expressions (e.g., splitting on white space and punctuations). However, for other types of data, such as product descriptions, medical prescriptions, and OCR text, the problem of tokenization is non-trivial and is as difficult as extracting the entities itself. For example, consider the following product descriptions from a grocer's inventory database:

COCA COLA CHERRY 12x12oz
COCA COLA CHERRY 12 12 oz
COCA_COLA CHERRY3LTR

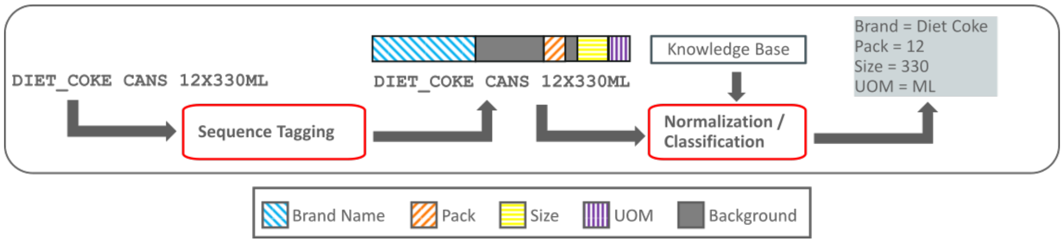

Figure 1: Example task, and legend for all figures

Suppose the goal is to extract the field labels such as brand(B), pack(P), size(S), and unit of measurement(U), and we want to ignore the rest (O). The first problem is that the unit of measurement(uom) often occurs immediately after the size (for example, “12oz” and “3LTR”). Therefore, the usual white space based tokenizer would fail to generate two separate tokens, which in turn will make it impossible for the sequence labeling model to assign correct labels. Of course, we could employ domain specific tokenization rules, but this is surprisingly difficult, because for example, a rule that correctly splits “3oz” may incorrectly split “7up.” Further, much of the data is entered in haste on a poor-quality computer terminal. Thus, the data contains numerous white-spacing errors, (e.g., “CHERRY3LTR”). Therefore, no single rule, or a small set of hand crafted rules would be sufficient to accurately tokenize the text.

Most rule-based approaches often employed in the industry for performing the task of attribute extraction are known to be fragile and expensive to develop per domain. In contrast, statistical methods are more robust to noise and adapting them to a new domain requires acquiring new labels, which is significantly less expensive compared to building new rules by a domain expert.

In this paper, we address the problem of attribute extraction by using a character-based model in which labels are associated with each character instead of each token. However, words still provide useful information in many situations. Thus, in addition to the raw characters, we use the original words from which the characters came (under some tokenization) as observed evidence for predicting the labels. For cases in which the tokenization is actually correct, these tokens will provide a strong signal for predicting the correct labels associated with the constituent characters. However, for cases in which tokenization is wrong, the character-level and Markov (transition) information can override the word.

Indeed, we demonstrate that a character-based model can achieve high accuracy on a number of product description datasets, reducing error by 48.1% over the traditional word-level baseline. It remarkably achieves almost the same accuracy as a word tokenizer that uses ground-truth manually tokenized text. To the best of our knowledge, using character observations as variables in a probabilistic sequence tagging model in this context has not been proposed in the literature, and can have applications in other fields with a lot of noisy data, eg. medical prescriptions.

The attributes extracted from noisy text using this model may contain spelling errors, abbreviations and other non-standard forms. In this paper, we also propose the use of a pair-wise entity-linking system to deal with this problem. We evaluate the character based sequence labeling models in isolation as well as in the context of a full end-to-end system present an end-to-end system for attribute extraction, along with multiple techniques for normalization.

2 Problem Setting

We are given a collection of short, noisy text descriptions and a list of attributes that we are interested in extracting. The goal of the system is to find the set of attribute value pairs from this text.

Consider the following example of an attribute extraction problem. Let us say we are given a product description, *DIET_COKE 6X200ml*. The attributes we need to extract are: brand name, pack size, size of individual unit, and the unit of measurement. The values that need to be ultimately predicted are: brand=*Diet Coke*, pack=*6*, size=*200*, uom=*ml*. Note that the desired value for the brand attribute is the cleaned and normalized version of what is observed in the text. We call the set of values that each attribute can take the domain of that attribute. For example, units of measurement can only take values from a fixed set of values like *ml*, *l*, *g*, *kg*, etc.

Often, these values are present explicitly in the text, and can therefore be extracted using sequence labeling. Inconsistent spacing is an endemic problem in many of these datasets. However, there are other issues with the text that must also be addressed. The text may contain noise, such as abbreviations, spelling errors or non-standard punctuation, e.g., *DIET_COEK*. Sometimes, the values are missing and must be inferred from other indirect evidence, such as the existing text or values of other attributes. For example, if the product description says, *DIET COKE 6X200*, then we might be able to infer that the unit of measurement is *ml*. In some cases, the domain of the values an attribute takes is closed. In others, the domain can be open and can change with time. One example is size, which can take arbitrary values like *260 (ml)*, and even though you could request the retailer to provide a list of all possible sizes, it might change quickly with time or due to ongoing promotions.

One possible way to extract values of attributes with a fixed domain is to treat it as a multi-class classification problem, with attribute value as a class label, and lexical and other attribute values as features. This would be a valuable strategy for cases when attribute values are themselves missing from the text. It would also work reasonably well for cases in which the attribute domain is small, and can therefore be learned using a reasonably small amount of training data. However, if the number of possible values an attribute can take is quite large (possibly even larger than the number of training examples), we need a different strategy. We will discuss one such strategy that uses a combination of sequence tagging and entity linking in the next section.

3 Method

3.1 System Architecture

We now describe our sequence labeling approach and the end-to-end system in which it operates. We assume that we are given the list of attributes for which we need to extract values. The end-to-end system for determining values for the given attributes consists of two parts: attribute sequence tagging and attribute normalization.

Figure 1 shows how a raw piece of text is first tagged with attribute values, which are then linked to a knowledge base to derive a normalized value.

3.2 Attribute Sequence Tagging

The core component of the attribute extraction system is sequence tagging, which tags the text with attribute labels. Given a piece of text, it annotates each token with labels for the attribute value that it is a part of. Conditional Random Fields (CRF) [5] have been used for a variety of entity extraction tasks [6, 9, 13]. In this section, we describe a novel approach for attribute sequence tagging that uses character-level tokens, instead of the usual word-level ones in a CRF.

3.2.1 Model

Typically, in a linear chain CRF used for entity extraction, each word token is considered an observation, and the corresponding hidden variable represents the state, or the label for that token. It is fairly common to use a BIO scheme to annotate attribute values in text, in which each token label is prefixed with either a 'B' to indicate the beginning of an entity name, or an 'I' to indicate the inside of an entity name, and 'O' is used to represent background tokens.

Under our proposed scheme, we use each character as an observation in the CRF, along with the BIO representation for the labels. The spaces between the words for an attribute value is considered to fall inside the value.

Figs. 2 and 3 show the difference between word-level and character-level CRF representation of two almost identical pieces of text. Notice how the word based model would not be able to assign different tags for "300" and "ML" if they were not separated by spaces.

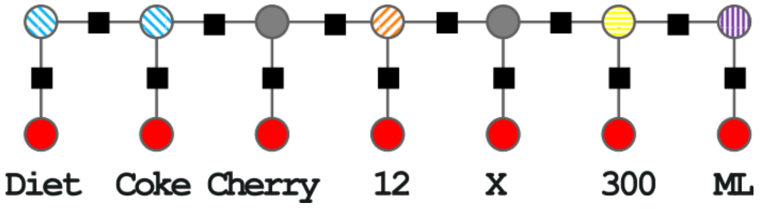

Figure 2: Word-level CRF

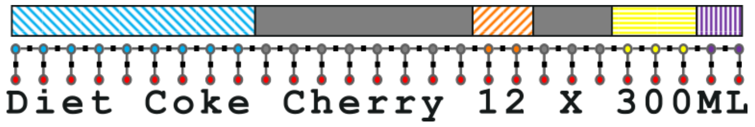

Figure 3: Character-level CRF

3.2.2 Features

In traditional linear chain CRF with word-level observations, lexical features, along with various properties of the word itself play an important role. In our new representation, characters by themselves do not carry as much information. The key to making character level sequence tagging work is to keep track of the original word from which a character came, and use the features of this original word, along with those of the character token itself. We simply tokenize the original text by space to keep track of the original words. It is important to note here that using features of the original words is fundamentally different from using words as observed variables in the CRF, because the original words are now only used as an evidence, and therefore the model is much more tolerant to the noise in tokenization. For example, the word-token feature for “200ml” narrows down the label space to essentially (S)ize and (U)oM, then the character level tokens help break the ties, e.g. “2” is more likely to be a size than a uom.

Some of the features used for this task are similar to those used for standard named entity recognition tasks. We employ the FACTORIE [8] toolkit’s ConllChainNER model as a starting point for our features. We use its features for both character and the original word, such as lowercased token, shape, and punctuation. We also use existence in lexicons and features from surrounding tokens. In addition, we consider new features relevant to this domain, such as size of the original word and position of the character relative to the start of the document.

3.2.3 Data Annotation Scheme

The CRF model uses BIO representation for labels, but for the purpose of easier annotation, we assume that no two attribute values of the same type occur next to each other in text. This assumption holds well in most product descriptions. We then represent each attribute type label by a single letter, and use ‘O’ for the background label. The resulting label string, placed right below the text greatly enhances the ease of annotation.

The annotation process involves three steps. First, a human assigns the true values of each attribute. Next, we match these attribute value strings to the description string automatically to mark token spans that represent them. Note that this is an approximate process, and we only employ this to reduce the manual effort for data annotation. In the third step, a human fixes the annotation string to generate the final version in which every character is a ground truth annotation.

3.3 Attribute Normalization

The attribute values extracted by the CRF may contain several inaccuracies, such as abbreviations, spelling errors, additional punctuation and so on. Most analytics applications that consume these attribute values require them to be normalized. This task can be challenging in the presence of ambiguities and missing information. It is also not desirable to write a lot of rules to normalize the extracted text, since there can be several different variations of the same underlying value, making the rules difficult to maintain.

3.3.1 Normalization as Linking

In many situations, we are provided with a standardized knowledge base of a set of values that an attribute can take. For example, even though the brand names found in a large corpus of product description are noisy, a small curated list of all the brands being offered by a retailer may be readily available. One way that we can exploit this information is by linking the raw, noisy 'mentions' of attribute values found in unstructured text to its corresponding clean value in the target knowledge base. By confidently establishing such a match, we can then use the cleaner text as the value for that attribute.

3.3.2 Pairwise Linker

We use a classification based, pairwise approach to link the extracted text segments to a target knowledge base of attribute values. The pairwise method for linking works as follows. Let m_i be a text segment of attribute t predicted by the CRF model. Let C_t be the set of all possible values that attribute t can take. Then, we generate all pairs, (m_i, c_j), such that $c_j \in C_t$. We then build an SVM classifier [15] to predict $P(y_{i,j})$, that represents the probability that m_i and c_j represent the same attribute values. After classifying each such pair, we select the best target value c_j, such that $argmax_j P(y_{i,j})$

The features, x_{ij} include the following: exact match, match without punctuation, binned, normalized edit distance and lexical features (concatenated strings of the mention and the candidate). In many situations, lexical features are unavoidable, since there is not enough information in the product description to match the extracted string to the target attribute value.

3.3.3 Normalization as Classification

In some situations, even though the actual attribute value is missing from the product description string, it can be inferred by observing the entire text, as well as predicted values of other, related attributes. In such cases, if the domain of the attribute is closed, we can treat it as a document classification problem with character n-grams and other extracted values as features.

4 Related Work

Product attribute extraction is a relatively less studied problem in the literature. Some of the early work to apply modern machine learning techniques to this problem are found in Ghani et al. [2, 10]. They extract both attribute names and values, and assume a richer input text that actually contains the attribute names, whereas, in our case, the attribute value appear in isolation, without cue about the attribute names. Putthividhya et al. [11] were the first to apply a CRF to the problem, also showing that traditional NER features such as POS and capitalization may not be very useful in this problem context. However, they did not deal with the problem of unreliable spacing issues, and used a word level CRF representation. Joshi et al. [3] also use a word-based CRF for this task and apply word embeddings to improve accuracy. Mauge et al. [7] proposed an unsupervied approach for discovering property names and values, along with synonym discovery. Again, their problem setting is different from ours, in that we can assume which attributes are of interest for analytics purposes.

Zhang and Johnson [16] show that character based features can help generalize to new unseen words. Although it is now common practice to use character n-grams as features in NER systems; surprisingly, there has not been a tremendous amount of previous work that uses a character-based model. Recently, Chrupala et al. [1] also highlight the need for character-level modeling and use embeddings for text segmentation. Our approach can also be adapted for that task. The work of Klein et al [4] incorporates character-based dependencies in the model and demonstrates that these dependencies help generalization on words that have been observed in the training set. In their work they recommend deterministically enforcing constraints that all the characters corresponding to a single token receive the same label. In contrast, our work addresses a different setting in which tokens

dataset	**# documents**	**# chars**	**# words**	**% words w/ errors**	**% w/2**	**% w/3**	**% w/4+**
Cereals	338	13484	1892	20.9	19.3	1.6	0.1
Spirits	558	21448	3365	33.9	28.8	0.9	4.0
Poultry	560	21372	3137	16.6	15.6	0.9	0.1
Tea	205	7557	1175	19.3	18.5	0.7	0.0
Yogurt	1080	47282	6909	18.0	13.0	1.2	3.8
Beer	593	22523	3276	16.2	15.8	0.1	0.3
Beverages	632	21027	3307	19.7	10.8	0.7	8.1
Average	566	22099	3294	**20.7**	**17.4**	**0.9**	**2.3**

Table 1: Dataset summaries. The column "% words w/ errors" shows the percentage of total word-based tokens that contain two or more unique character labels. The columns labeled "% w/n" show the percentage of total tokens that contain exactly n unique character labels.

are not observed and must be inferred from the data. Tseng et al. [14] use CRFs for Chinese text segmentation, but Chinese characters are information dense leading to a different model behavior.

5 Experiments

5.1 Data

We evaluate our models on seven datasets in the grocery domain: tea, spirits (liquors), cereals, beverages (soft-drinks), yogurt and beer. Each category was labeled by one person from our team by following detailed annotation guidelines. It was difficult to get multiple annotations per label for 3966 SKUs (crowdsourcing was not used due to the proprietary nature of data). Target knowledge bases for entity linking were provided.

Each of these datasets contain the brand, product name, flavor, size, pack, and unit of measurement attributes. In addition, the spirits and beer datasets contain attributes for alcohol percentage. We summarize the datasets in Table 1. Of particular interest is the percentage of space separated word tokens that comprise multiple ground-truth attribute labels since this statistic indicates the extent to which tokenization is a problem. For these datasets, this tokenization problem is pervasive: 20.7% on average and as high as 33.9% for beer.

5.2 Systems

We compare the character-based model advocated in this paper to three word-based models for a total of four sequence labeling strategies.

- **Character:** the character-based model in which each character is treated as a token variable in a linear CRF.
- **Word:** the traditional word-based model in which the text is toknized into words and each word is associated with a label variable. We employ white-space tokenization which treats each sequence of non white-space as a word (token). Since the data is originally labeled at a character-level, it is possible that some tokens contain multiple character-level labels. For these tokens, we set the true token-level label to be the character-level label that appears most frequently in the token. For example, the token "100ml" contains three character labeled as "Size" and two character labeled as "UoM;" thus, we would set the token level label to be "Size" and not "UoM."
- **Word-expanded:** same tokenization as above, except we expand the label space to allow for multiple labels per token. In particular, we set the token-level label to be the set of character-level labels. For example, the correct label token "100ml" would be "Size/UoM."
- **Oracle-tokenization:** we employ the character-level labels to find a ground-truth tokenization in which no token contains multiple labels. Specifically, we employ the white-space tokenization and then split any tokens that contain multiple labels. For example, "200ml" would be split into two tokens "200" and "ml" with labels "Size" and "UoM" respectively.

	Character Level				End-to-End			
dataset	oracle-tok	character	word	word-exp	oracle-tok	character	word	word-exp
Cereals	91.1	87.1	**87.9**	84.4	83.0	**79.9**	64.5	63.6
Spirits	93.1	**93.3**	88.3	87.9	78.2	**77.5**	64.6	64.8
Poultry	96.9	**95.9**	92.5	89.7	82.7	**82.6**	63.6	63.5
Tea	93.4	**93.7**	89.3	86.0	92.2	**91.3**	77.0	75.9
Yogurt	91.1	**89.0**	87.0	83.5	81.4	**80.8**	61.9	61.5
Beer	91.9	**91.2**	82.3	78.3	67.8	**66.5**	47.1	47.2
Beverages	95.6	**94.9**	87.2	84.5	85.6	**85.3**	59.6	58.8

Table 2: Accuracy per Dataset. Note that oracle tokenization is not achievable in practice

	Sequence Labeling				End-to-end		
Method	Char Acc	Brand Seg. F1	Size Seg. F1	All Seg. F1	Brand Acc	Size Acc	All Acc
oracle-tokenization	0.933	0.895	0.992	0.909	66.7	93.3	81.2
character	0.922	0.852	0.974	0.877	65.6	92.0	80.6
word	0.878	-	-	-	59.7	11.0	62.6
word-expanded	0.849	-	-	-	59.3	11.9	62.2

Table 3: Combined results across seven datasets.

Note that this model cannot actually be employed in a real-world system since it requires labeled data. We only include the system for comparison.

We study each of these strategies both in isolation (i.e., as a sequence labeling problem) and in the context of the full end-to-end system. As mentioned in the Method section, running the full system is a two step process. First, we run a sequence labeler to identify attributes that are present in the text. Second, we run a normalization step that examines the text and sequence labeling predictions in order to arrive at a final set of attribute values. Specifically, we perform entity-linking to match extracted brands, product names and flavors to their canonical form, and run a classifier to predict the unit of measurement from the text and the labels. The remaining attributes (size, pack, alcohol percentage) do not undergo a normalization process; though we predict a default size and pack size of "1" if the sequence labeler does not extract values for them.

5.3 Evaluation

As mentioned earlier, we employ two types of evaluation, an end-to-end extraction evaluation which evaluates the final classification accuracy of the predicted attributes, and a sequence labeling evaluation which evaluates the sequence labeling component in isolation.

However, evaluating the sequence labeling components in isolation is non-trivial because they are not directly comparable under traditional sequence labeling evaluation metrics such as the segment-wise F1 employed in NER [12]. This is because the output space of the sequence labeling problem is different for each model; more specifically, (1) the class of labels is not always the same (e.g., the expanded model contains conjunctions of labels such as "Size-UoM" in its label space) and (2) the tokenization determines the number of labels.

We instead employ a character-level evaluation metric that is more suitable for direct comparisons: first, we convert each system's token-level predictions into character-level predictions and then compute a per-character accuracy. Thus, we have two functions to define: (1) the projection of word-level predictions onto character-level predictions and (2) the evaluation function that operates on character-level predictions. For the word-based systems, we project labels onto characters in the following way.

For each character c, either c is contained in some token t_i or c is a delimiter and not part of any token (and t_i is considered to be null). Let t_{i-1} be the previous token in the sequence and t_{i+1} be the next token. Let l_i, l_{i-1}, l_{i+1} be the predicted labels associated with each token. If c is a member of a token, then the projection is to simply employ the label l_i; however, if c is not a member of a token then we must rely on neighboring labels. More precisely, we define the predicted character label l_c as follows:

$$l_c = \begin{cases} l_i & \text{if } t_i \neq NULL \\ l_{i-1} & \text{if } t_i = NULL \wedge l_{i-1} = l_{i+1} \\ O & \text{if } t_i = NULL \wedge l_{i-1} \neq l_{i+1} \end{cases} \tag{1}$$

Although the projection step allows us to compare systems with different tokenizations, we still must cope with the problem that the output (label) space of the systems differ. Specifically, we must design our evaluation metric to handle cases in which the word-expanded system predicts that a token contains multiple labels. In order to err on the side of a stronger baseline system, we give the baseline system full credit if any one of its predictions is correct for a particular token. For example, if the original token is "200ml" (with character labels "SSSUU") and the word-expanded system correctly predicts the label "SU," then the system will receive 100% accuracy for all the characters in this token. If the system had instead predicted "S," the accuracy would instead be $3/5 \times 100 = 60\%$, which is full credit for "S."

For the end-to-end evaluation, we run the full system including the sequence labeling and normalization as described before. We then measure end-to-end accuracy as the proportion of correctly predicted final value for the attribute.

5.4 Results

We trained a separate model for each domain of the grocery data, and defer the study of cross-domain generalizability to future work. All results reported in this section are obtained by averaging over five runs of 5-fold cross validation, with three folds for training the extractor, one for training the entity linker, and one for testing.

In Table 2 we evaluate the performance of the four systems on each of the seven datasets. We report the character level accuracy for the sequence tagging task, as well as the end-to-end accuracy after entity linking. For more detailed results per dataset, please refer to the supplementary material. In Table 3 we show the results combined across all seven datasets. For completeness, we also include the segment-wise F1, but please note that this metric cannot be reliably calculated for two of the baseline tokenization approaches. We also report results on two of the attribute types to illustrate the effectiveness of our approach on a variety of attribute types.

We observe that the character based system consistently and substantially outperforms both the word and word-expanded approaches (reducing average error across datasets by 36.1% in sequence labeling and 48.1% in the end-to-end). Note that attributes such as brands are more challenging to extract, compared to a more regular attribute like size (average end-to-end accuracy 92%) . However, this is also an example of the type of attribute where the word based approach essentially fails (average end-to-end accuracy 11%). Some product categories are also more challenging. For example, in case of Beer, we found that most of the brand names did not exist in the product description at all, and had to be inferred purely based on external information to which the system did not have access.

Second, we observe that the character based model consistently performs competitively with the oracle-tokenization system. This is an especially encouraging result because it demonstrates that the character-based model is able to perform as if it had access to perfect tokenization.

6 Conclusion and Future Work

We proposed a character-based sequence labeling CRF model for joint tokenization and tagging of attribute values in noisy text. We showed that it performs better than using words as observations for text with inconsistent spacing. We also proposed using pairwise entity linking for normalizing the extracted values and show that an end-to-end system with these two components can be extremely effective for attribute extraction in practice. We compare our approach with multiple baselines on real-world product description data and show the viability of our solution. In the future, we would like to exploit information across several data points for improved normalization. It would also be interesting to study how well such approaches generalize across multiple categories and domains.

7 Acknowledgements

We would like to thank our team at Oracle Labs for useful discussions. We would also like to thank our partner business unit members to help with data annotations.

References

[1] Grzegorz Chrupala. Text segmentation with character-level text embeddings. *arXiv preprint arXiv:1309.4628*, 2013.

[2] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. Text mining for product attribute extraction. *SIGKDD Explor. Newsl.*, 8(1):41–48, June 2006.

[3] Mahesh Joshi, Ethan Hart, Mirko Vogel, and Jean-David Ruvini. Distributed word representations improve ner for e-commerce. In *Proceedings of NAACL-HLT*, pages 160–167, 2015.

[4] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 180–183, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[5] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, 2001.

[6] Wei Li and Andrew McCallum. Rapid development of hindi named entity recognition using conditional random fields and feature induction. *ACM Trans. on Computational Logic*, 2(3):290–294, September 2003.

[7] Karin Mauge, Khash Rohanimanesh, and Jean-David Ruvini. Structuring e-commerce inventory. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 805–814, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[8] Andrew McCallum, Karl Schultz, and Sameer Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009.

[9] Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Inf. Process. Manage.*, 42(4):963–979, July 2006.

[10] Katharina Probst, Rayid Ghani, Marko Krema, Andrew Fano, and Yan Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2838–2843, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[11] Duangmanee (Pew) Putthividhya and Junling Hu. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1557–1567, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[12] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *In Proceedings of Computational Natural Language Learning*, pages 142–147. ACL Press, 2003.

[13] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267373, 2012.

[14] Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171, 2005.

[15] Vladimir Naumovich Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[16] Tong Zhang and David Johnson. A robust risk minimization based named entity recognition system. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 204–207, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.