Neural Network based Extreme Classification and Similarity Models for Product Matching

Kashif Shah, Selcuk Kopru, Jean-David Ruvini
eBay Research, Hamilton Avenue San Jose, CA 95125, USA
{skshah,skopru,jean-david.ruvini}@ebay.com

Abstract

Matching a seller listed item to an appropriate product has become a fundamental and one of the most significant step for e-commerce platforms for product based experience. It has a huge impact on making the search effective, search engine optimization, providing product reviews and product price estimation etc. along with many other advantages for a better user experience. As significant and vital it has become, the challenge to tackle the complexity has become huge with the exponential growth of individual and business sellers trading millions of products everyday. We explored two approaches; classification based on shallow neural network and similarity based on deep siamese network. These models outperform the baseline by more than 5% in term of accuracy and are capable of extremely efficient training and inference.

1 Introduction

E-commerce marketplaces such as Amazon and eBay provide platforms where millions of people trade online every day. The growth of such marketplaces has been exponential in recent years with millions of active sellers and billions of live items listed at any point in time, bringing new interesting challenges.

On the buying side, while it provides buyers with more options and flexibility, it renders searching for a specific item and comparing items difficult. On the selling side, it dilutes the visibility of the sellers' items both on the marketplace and on external search engines such as Google and Bing. Also, it requires the sellers to provide a lot of details about the goods they are listing.

The product based experience is an enticing search experience that helps fulfill both buyers and sellers needs. The buyers can easily search, compare and make a decision on the product they wish to purchase while the sellers can speed up their listing process by using product details and stock photos from existing product catalog resulting in a professional-looking listing, more appealing in search results.

The process of automatically mapping items[1] to products become critical. Different sellers may describe the same item in very different ways, using very different terminologies. While some business sellers provide rich, structured description of their listings, the vast majority only provide short and sometimes insufficient information. The problem become even harder when information provided is not specific enough to identify the corresponding products. Lots of studies have been focused on structuring the products inventory by extracting and enriching attribute-value pairs (Mauge et al., 2012; Ghani et al., 2006) from various sources, and feed them into a matching function. The direction towards learning automatic semantic relationships on unstructured sequence of e-commerce text has been of less focus.

Neural based methods have recently shown to work well to capture the meanings and semantic relationships among words in textual data. In this paper, we aim at studying such approaches for e-commerce domain. We explored two neural network architectures for our product matching task; a simple shallow neural network model based on the *fastText* (Joulin et al., 2016b; Bojanowski et al., 2016; Joulin et al., 2016a) library and a deep siamese network model based on bidirectional long short term memory (Neculoiu et al., 2016; Mueller and Thyagarajan, 2016). We approach the problem in two ways, as a classification task where we have access to item listings and

[1] An *item* is an offer from a seller for a specific good containing seller specific information (description, condition, return, etc.). A *product* is the catalog (manufacturer) description of a good.

Proceedings of NAACL-HLT 2018, pages 8–15
New Orleans, Louisiana, June 1 - 6, 2018.

corresponding product ids, and as a similarity task where we also have access to product information. These approaches constitute different objective functions; classification aims at classifying an input instance into the identity classes, while similarity objective is to minimize the intra-class distance while maximizing the inter-class distance.

In the next section, we will discuss the challenges, section 3 contains the approaches and models we explored, and section 4 discusses the experiments and results. We will conclude after briefly reviewing the related work in section 5.

2 Challenges

Product matching is a difficult task due to the wider spectrum of products, many alike but different products, often missing or wrong values, and frequent variation in textual nature of products. In the following sections, we will briefly cover the challenges that make the task of product matching hard.

2.1 Data Sparsity

While some business and professional sellers provide rich information of their item listings, a vast majority of medium and small sellers provide short, unstructured and sometimes incomplete descriptions. This results in a lot of data sparsity due to missing values that are important to identify the products.

2.2 Product Duplicates

Data sparsity and erroneous information automatically extracted from unstructured or semi-structured sources may result into creation of the same product twice or multiple times called duplicates. The product duplicates are not only one of the biggest source of bad product experience, but also makes the product matching harder as multiple entities of the same product exists with overlapping information. The duplicates can have many fatal effects, including preventing machine learning algorithms from discovering important consistencies in product representations.

2.3 Single Source Products

There are many rare products such as antiques, collectibles, books, sculptures, etc. that are offered by only a particular single or business seller. Such products are often referred as single source products. In such cases, the description of the product can not be cross-validated in the absence of alternative sources. This causes a *one-shot* learning problem, which consists of learning a class from a single labeled example.

2.4 Product Bundles and Lots

Bundles are defined as multiple different products being grouped together and sold as a single offer. If the offer contains multiples of the same product these offers are referred as a *lot*. They make data ambiguous such as a number in product description could be a lot quantity or variation of product edition. Such product offers exhibit another level of complexity requiring special treatment or a separate model to identify lots and bundles.

2.5 Extreme Classification

Traditionally, multi-class classification solutions have been scalable up-to few hundred classes. As the number of classes grow, the time and space complexity increase exponentially to the extent that standard models do not scale up. For the task of product matching, the number of products are in millions and hence the methods that can scale up to million classes are required.

2.6 Open Set Classification

The number of classes are usually pre-defined for classification tasks and models are designed to predict one of those pre-defined classes for a given instance. For the task for product matching, new products are created everyday and they become part of inventory incrementing the amount of products daily. It either requires to re-train the models frequently with newly created products with additional classes or to employ models capable of handling unseen products.

3 Text Classification vs Similarity

The aim of text classification is to assign some piece of text (source) to one or more predefined classes or labels (targets). The piece of text could be a document, news article, listing offers, email, product review etc. Depending upon the task, the target labels are usually topic, category, product, sentiment id or name. Traditional classification approaches are employed to learn feature representation on the source and attempt to predict the target label. There are cases where inputs are text pairs along with their relevance score such as question-answer, query-document, source-

translation pairs etc. Both input texts can be encoded with rich representations. For such cases similarity based methods are natural architecture to learn differentiation among them.

We research both classification and similarity algorithms for product matching. Classification based approach using shallow neural network based on *fastText* library turns out to be very efficient with good performance but it requires to re-train the models for newly created products as mentioned in previous section. The similarity based method does not have this bottle neck as it is tailored towards open class set problem given a sufficient amount of paired dataset. The similarity based method also allows to use product side data in more natural way to model our paired item-product data.

3.1 Shallow Neural Network - *fastText*

Learning word representations has a long history in natural language processing. These representations are typically learned from large corpora either using count statistics or distributed semantic representations. Recently, (Mikolov et al., 2013) proposed simple log-bilinear models to learn continuous representations of words on very large corpora efficiently using Continuous Bag Of Words (CBOW) and Skip-gram models. Both of these are shallow neural networks which map word(s) to the target variable which is also a word(s). Both of these methods learn weights which act as word vector representations. The assumption of these models is that semantically or grammatically related words are mapped to similar geometric locations in a high-dimensional continuous space. The probability distribution is thus much smoother and therefore the model has a better generalization power on unseen events.

Recently proposed *fastText* (Joulin et al., 2016b)[2] library extended these models for supervised classification tasks. This architecture is similar to the CBOW model of (Mikolov et al., 2013), where the center word is replaced by a label. They also incorporated bag-of-ngram as additional features to capture some partial information about the local word order. The n-grams features are efficiently implemented by using the hashing trick (Goodman, 2001).

The output layer consists of a softmax function that assigns a probability distribution to each of the target label. The loss function aims at minimizing the negative log-likelihood over the classes. Because full-softmax is not scalable for large number of classes, *fastText* offers hierarchical-softmax as an alternate approach. This implementation uses huffman tree to reduce the complexity to logarithmic while incurring minor performance degradation. The hierarchical softmax function is the key factor that makes the task scalable to our million class dataset. For full details, we refer the reader to the original paper (Joulin et al., 2016b).

For our product matching task, these models not only offer the benefit of the rich distributed representation of item listings but also allow to deal with millions of classes while reducing the training time from days to minutes, and keeping good performance in terms of accuracy.

3.2 Siamese Networks

Siamese networks are a special type of neural network architecture having two identical structure trained simultaneously with shared weights. Instead of a model learning to classify its inputs, the neural network learns to differentiate between two inputs. In our case, we are interested in finding similarity between the item listings and corresponding products given pairs whose semantic similarity has been labeled. Our siamese network consists of two parallel neural networks, each taking one of the two inputs, listings or products in our case, along with a binary score (1 for match and 0 for mismatch).

Different kind of parallel structures has been explored for various tasks including Convolutional Neural Networks (CNN), Skip-thought vectors, Tree-LSTM etc. Recurrent Neural Networks (RNN) have shown promising performance on textual sequence of data for various NLP tasks. The Long Short Term Memory (LSTM) networks proved to be superior to basic RNNs for learning long range dependencies through its use of memory cell units that can store/access information across lengthy input sequences. The Bidirectional LSTM networks can leverage the knowledge present on both sides of the current word to encode the text and have shown good performance on various NLP tasks like named entity recognition (Huang et al., 2015; Wang et al., 2015) and text similarity (Neculoiu et al., 2016).

In this work, we studied BiLSTMs for our siamese network architecture that is very similar

[2]https://github.com/facebookresearch/fastText

to (Mueller and Thyagarajan, 2016) in architecture but we employed BiLSTM along with the contrastive loss function (instead of Manhattan LSTM) [3]. The network consists of three hidden layers and each layer consists of 50 hidden units. The LSTM units of our bidirectional network contain a memory state having an input, forget and output gate with the logistic function on the gates and the hyperbolic tangent (tanh) on the activations:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
j_t &= \tanh(W_{xj}x_t + W_{hj}h_{t-1} + b_j) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes j_t \\
h_t &= \tanh(c_t) \otimes o_t
\end{aligned}
$$

where σ is the logistic sigmoid function, $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho}, \mathbf{W}_{hj}$ are recurrent weight matrices and $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo}, \mathbf{W}_{xj}$ are projection matrices. $\mathbf{b}$ are the bias terms in equations.

The inputs are represented as a sequence of words with a maximum length of 100 and (shorter sequences are padded). The output of the combined model at each time step is simply the concatenation of the outputs from the forward and backward networks. The outputs of the last layer are averaged over time and the output vectors are fed to the contrastive loss function:

$$E = \frac{1}{2N}\sum_{n=1}^{N}(y)\,D + (1-y)\max(m-d, 0)$$

where $D = ||a_n - b_n||_2^2$ is euclidean distance between the outputs a_n and b_n of the two parallel networks. In our case, y is either 1 or 0. If the inputs are from the same class, then the value of y is 1, otherwise y is 0. $m > 0$ denotes a margin which acts as a boundary (with the radius m). The motivation behind this loss function is that similar pairs tend to be as close as possible and dissimilar pairs must be separated by a distance defined by the margin. The network is optimized using Adam optimizer along with dropout to prevent over-fitting.

4 Experiments and Results

We performed a set of experiments on an *eBay* dataset. The data has been collected over the years and contain millions of products and billions of listings. We divided the data into meta-categories and models are trained on these categories separately. This partition into categories not only keep the data size to a affordable level given the resources but also allow us to train the models in parallel for each category being independent.

In this paper, we only report the experiments for three categories namely Electronics, Clothes, Shoes and Accessories (CSA), and Collectibles as the proof-of-concept. In the next sections, we will briefly explain how this historical data has been created.

4.1 Dataset

Many branded items have unique identifiers that help buyers recognize and find them, including the items *brand*, Manufacturer Part Number (*mpn*), and Global Trade Item Number (*gtin*). The *gtin* can include an items Universal Product Code (*upc*), European Article Number (*ean*), or International Standard Book Number (*isbn*). Around 80% of our training data is created with these identifiers. The remaining 20% data is either adopted[4] by sellers or made available from third party sources. The statistics about the data used for experiments reported in the paper are given in table 1.

Our dataset consists of four fields, the *title* that are available for all instances, *gtin*, *brand* and *mpn* that are partially available as shown in Table 1. The classification experiments are based on combination of these inputs for item listings along with *product-id* as label. The inputs are just concatenated to make a single sequence of text. For similarity experiments, we made use of corresponding product *title*, *gtin*, *brand* and *mpn*. For the test set, we sampled the data randomly from our training set and performed manual validation. The validation is done against multiple sources available online such as other marketplaces etc. The validation score is simply a 1/0 label, 1 if the product against the item listing is correct, otherwise 0.

Basic data normalization is performed such as tokenization, lower casing, removing noise words (e.g. doesn't apply, not applicable etc). We removed the instances that contain a single word as they do not convey any meaningful information. Exact same listings are merged into a single in-

[3] we adapted an open source implementation for our task: https://github.com/dhwajraj/deep-siamese-text-similarity

[4] The sellers chose among the existing products in the catalog for their item listings.

Category	#listings	gtin	brand	mpn	#products	test
Electronics	9.38	5.60	7.19	6.45	3.17	6468
CSA	21.09	6.73	11.96	3.75	9.78	3549
Collectibles	0.80	0.28	0.58	0.38	0.58	1999

Table 1: Data Statistics (all numbers are in millions except the test set). *gtin*, *brand* and *mpn* column show the total number of these fields available for item listings.

		fastText				Siamese
Category	baseline	title	title+gtin	title+brand+mpn	title+gtin+brand+mpn	all
Electronics	84.4	81.1	86.2	85.9	87.9	**89.9**
CSA	75.7	73.9	76.7	76.2	78.8	**82.4**
Collectibles	92.8	89.3	94.5	93.7	96.4	**97.1**

Table 2: Accuracy for 3 Meta-Categories: Electronics, CSA, and Collectibles

Category	pre-emb	product-data	accuracy
Electronics	yes	-	88.4
	-	yes	88.2
	yes	yes	**88.8**

Table 3: Accuracy with *fastText* models for Electronics using pre-embeddings and additional product data.

stance.

4.2 Results

Our baseline models are based on a waterfall approach. The first two modules are based on lookup tables for unique identifiers. In a first module, if the item listing has a *gtin* and it matches with the product in the inventory, then the item is adopted to matching product. The second module is based on *brand-mpn* lookup tables. Finally, if none of the unique identifiers are available, then the item listing is adopted to a product with highest string match[5].

We trained many *fastText* models with various combination of inputs. For all of our models, the embedding dimension is set 300, learning rate to 0.5, and number of epochs to 10. The n-gram parameter is set to 3 and it brings nice improvements than the models without these additional features. The hierarchical-softmax has been used as a loss function.

The results are shown in table 2. The combination of features produce better results than the models trained with *title* only even though *gtin*, *brand* and *mpn* are partially available. The combination of all features give the best performance followed by models with combination of *title+gtin*. This is because of the fact that the *brand* or *mpn* are sometimes already available in *title* while *gtin* rarely appear in listing *title*. Therefore, inclusion of additional *brand* and *mpn* do not bring as much additional information as *gtin*.

We also trained the *fastText* models using pre-trained embedding and product data as additional resource for Electronics. The pre-embeddings are learned with *fastText* using unsupervised CBOW model on all of the listings available for the respective category. In a second setting, we simply merge the product data in our training set. The results are shown in table 3. Even though our initial dataset is quite large but it is evident from the results that these additional resources bring further improvements as they offer more coverage and completeness.

For Siamese Network, the models are trained using the item-product pairs as described in section 3.2. We set the embedding dimensions to 300, number of epochs to 100 and mini-batch size to 128. During inference for a given item listing, we first need to generate the item-product pairs to measure the similarity. It is prohibitively impractical to compare each item with all the products in the given category particularly for inference during production. Therefore, we first reduced the target space by generating top 500-best products with *fastText* models. Then, these pairs are fed to the Siamese network to calculate the similarity. The pair having highest similarity score is picked as matching. We also explored the n-gram matching approach to reduce the target space but n-best with *fastText* method turns out to be bet-

[5] any unigram overlap.

Category	*fastText* train	*fastText* infer	Siamese train	Siamese infer
Electronics	56s	5ms	20mins	15ms
CSA	90s	8ms	30mins	21ms
Collectibles	20s	3ms	10mins	11ms

Table 4: *fastText* versus Siamese training (per epoch) and inference time (per instance), mins=minutes, s=seconds, ms=milliseconds

ter. Siamese models show the same trend as with *fastText* models on different combination of inputs showing best performance when using all of them (the last column in table 2).

In table 4, we present the training (per epoch) and inference time (per instance) for both kind of models[6]. The *fastText* models are extremely efficient to train as compared to the Siamese network. These classification models can be trained everyday to cope with open class set problem as mentioned in section 2.6. Siamese network though take longer training time, but they do not need everyday re-training and shown to be better in term of performance.

4.3 Product Deduplication

Our product matching method resulted into an another interesting findings for product deduplication. The *fastText* models produce the probability score along with predicted product for a given item listing. The probability score encodes the confidence score for the predicted label. We have collected the instances in our test set where the confidence score is very high (i.e closer to 1) and predictions do not agree with the labels in our sampled test set. Manual inspection of this data revealed that many predictions are actually correctly adopted to another duplicate product in the catalog.

For these experiments, we manually evaluated the test set for the Electronics category and sampled the data based on disagreement where the confidence score is equal or higher than 0.99. This yields 2% of the instances of our test set. The result showed 89% accuracy demonstrating the potential of using confidence score for product deduplication. As we drop the threshold, the coverage increases but accuracy starts to degrade as expected. We plan to further investigate this in our future work along with Siamese Network based product deduplication on inter-product similarity score.

[6]All *fastText* models are trained on Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz and all Siamese models are trained on 4 Nvidia GPUs GM200 [GeForce GTX TITAN X] with 64GB total memory.

5 Related Work

The problem of product matching has been largely focused on attribute-value pair extraction to retrieve a unique set of attributes or in other words structuring the e-commerce inventory (Mauge et al., 2012; Ghani et al., 2006). The attribute and feature extraction studies go from regular expression (Köpcke et al., 2012) to named entity recognition methods based on CRF models (Melli, 2014; Ristoski and Mika, 2016). There are wide range of matching algorithms studied from various string matching (Vandic et al., 2012; Thor, 2010) to more advance methods. We refer the reader to Ristoskia et al. (2017) and Kannan et al. (2011) for a good brief overview of these studies.

Many of these methods aim to convert the free text into a structured representation and employ matching models by weighting the attributes based on their significance. In this paper, we do not model explicitly to first structure the offers and products. In our case, both offers and products are sequence of free text, and we let the models learn representations based on large historical data. Vector based models such as word2vec (Mikolov et al., 2013), glove (Pennington et al., 2014) and skip-thought (Kiros et al., 2015) have shown promising results on textual data to learn semantic representations. Sequence models based on neural networks is a hot topic in the NLP community and variety of models have been researched for different kind of tasks including more recently convolutional and recurrent networks (RNN). Convolutional networks based siamese network have been successful to find image similarity (Chopra et al., 2005; Koch, 2015). Many variants of siamese recurrent neural networks have been studies on textual sequences including manhattan-LSTM (Mueller and Thyagarajan, 2016) and bidirectional LSTM (Neculoiu et al., 2016). Our work takes the motivation from these siamese LSTM models and studied them on e-commerce domain for product matching task along with *fastText* models for extreme classification using hierarchical softmax.

6 Conclusion

We have proposed classification and similarity based approaches for product matching task. The inputs to our models are sequence of texts unlike many of the previous studies where more focus has been on structured text for such tasks. The classification approach based on the *fastText* models can scale up to millions of classes, outperforms the baseline and is extremely efficient to train. The similarity approach based on siamese network makes use of both item listings and product text. They not only improves the accuracy further but also have the advantage of avoiding frequent retraining. As a by-product, we also have shown the potential to use our models for product deduplication, which we plan to explore further in our future work. Finally, our models can accommodate any information (beyond *title*, *gtin*, *brand* and *mpn*) easily amenable into a textual form. We are now planning to extend them with image data.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 561–564. IEEE.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016a. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Anitha Kannan, Inmar E Givoni, Rakesh Agrawal, and Ariel Fuxman. 2011. Matching unstructured product offers to structured product specifications. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 404–412. ACM.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*.

Gregory Koch. 2015. *Siamese Neural Networks for One-Shot Image Recognition*. Ph.D. thesis, University of Toronto.

Hanna Köpcke, Andreas Thor, Stefan Thomas, and Erhard Rahm. 2012. Tailoring entity resolution for matching product offers. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 545–550. ACM.

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

Gabor Melli. 2014. Shallow semantic parsing of product offering titles (for better automatic hyperlink insertion). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1670–1678. ACM.

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

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Petar Ristoski and Peter Mika. 2016. Enriching product ads with metadata from html annotations. In *International Semantic Web Conference*, pages 151–167. Springer.

Petar Ristoskia, Petar Petrovskia, Peter Mikab, and Heiko Paulheima. 2017. A machine learning approach for product matching and categorization. *Semantic web*.

Andreas Thor. 2010. Toward an adaptive string similarity measure for matching product offers. In *GI Jahrestagung (1)*, pages 702–710.

Damir Vandic, Jan-Willem Van Dam, and Flavius Frasincar. 2012. Faceted product search powered by the semantic web. *Decision Support Systems*, 53(3):425–437.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*.