Chinese Zero Pronoun Resolution with Deep Neural Networks

Chen Chen and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{yzcchen,vince}@hlt.utdallas.edu

Abstract

While unsupervised anaphoric zero pronoun (AZP) resolvers have recently been shown to rival their supervised counterparts in performance, it is relatively difficult to scale them up to reach the next level of performance due to the large amount of feature engineering efforts involved and their ineffectiveness in exploiting lexical features. To address these weaknesses, we propose a supervised approach to AZP resolution based on deep neural networks, taking advantage of their ability to learn useful task-specific representations and effectively exploit lexical features via word embeddings. Our approach achieves state-of-the-art performance when resolving the Chinese AZPs in the OntoNotes corpus.

1 Introduction

A zero pronoun (ZP) is a gap in a sentence that is found when a phonetically null form is used to refer to a real-world entity. An anaphoric zero pronoun (AZP) is a ZP that corefers with one or more preceding mentions in the associated text. Below is an example taken from the Chinese Treebank (CTB), where the ZP (denoted as *pro*) refers to 俄罗斯 (Russia).

[俄罗斯] 作为米洛舍夫维奇一贯的支持者，*pro* 曾经提出调停这场政治危机。

([Russia] is a consistent supporter of Milošević, *pro* has proposed to mediate the political crisis.)

As we can see, ZPs lack grammatical attributes that are useful for overt pronoun resolution such as number and gender. This makes ZP resolution more challenging than overt pronoun resolution.

Automatic ZP resolution is typically composed of two steps. The first step, AZP identification, involves extracting ZPs that are anaphoric. The second step, AZP resolution, aims to identify an antecedent of an AZP. State-of-the-art ZP resolvers have tackled both of these steps in a supervised manner, training one classifier for AZP identification and another for AZP resolution (e.g., Zhao and Ng (2007), Kong and Zhou (2010)).

More recently, Chen and Ng (2014b; 2015) have proposed unsupervised probabilistic AZP resolution models (henceforth the CN14 model and the CN15 model, respectively) that rival their supervised counterparts in performance. An appealing aspect of these unsupervised models is that their language-independent generative process enables them to be applied to languages where data annotated with ZP links are not readily available. Though achieving state-of-the-art performance, these models have several weaknesses.

First, a lot of manual efforts need to be spent on engineering the features for generative probabilistic models, as these models are sensitive to the choice of features. For instance, having features that are (partially) dependent on each other could harm model performance. Second, in the absence of labeled data, it is difficult, though not impossible, for these models to profitably employ lexical features (e.g., word pairs, syntactic patterns involving words), as determining which lexical features are useful and how to combine the potentially large number of lexical features in an unsupervised manner is a very challenging task. In fact, the unsupervised models proposed by Chen and Ng (2014b; 2015) are unlexicalized, presumably owing to the aforementioned reasons. Unfortunately, as shown in previous work (e.g, Zhao and Ng (2007), Chen and Ng (2013)), the use of lexical features contributed significantly to the performance of state-of-the-art supervised AZP resolvers. Finally, owing to the lack of labeled data, the model parameters are learned to maximize data

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 778–788,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics

likelihood, which may not correlate well with the desired evaluation measure (i.e., F-score). Hence, while unsupervised resolvers have achieved state-of-the-art performance, these weaknesses together suggest that it is very challenging to scale these models up so that they can achieve the next level of performance.

Our goal in this paper is to improve the state of the art in AZP resolution. Motivated by the aforementioned weaknesses, we propose a novel approach to AZP resolution using deep neural networks, which we believe has three key advantages over competing unsupervised counterparts.

First, deep neural networks are particularly good at discovering hidden structures from the input data and learning task-specific representations via successive transformations of the input vectors, where different layers of a network correspond to different levels of abstractions that are useful for the target task. For the task of AZP resolution, this is desirable. Traditionally, it is difficult to correctly resolve an AZP if its context is lexically different from its antecedent's context. This is especially the case for unsupervised resolvers. In contrast, a deep network can handle difficult cases like this via learning representations that make lexically different contexts look similar.

Second, we train our deep network in a supervised manner.[1] In particular, motivated by recent successes of applying the mention-ranking model (Denis and Baldridge, 2008) to entity coreference resolution (e.g., Chang et al. (2013), Durrett and Klein (2013), Clark and Manning (2015), Martschat and Strube (2015), Wiseman et al. (2015)), we propose to employ a ranking-based deep network, which is trained to assign the highest probability to the correct antecedent of an AZP given a set of candidate antecedents. This contrasts with existing supervised AZP resolvers, all of which are classification-based. Optimizing this objective function is better than maximizing data likelihood, as the former is more tightly coupled with the desired evaluation metric (F-score) than the latter.

Finally, given that our network is trained in a supervised manner, we can extensively employ lexical features and use them in combination with other types of features that have been shown to be useful for AZP resolution. However, rather than employing words directly as features, we employ word embeddings trained in an unsupervised manner. The goal of the deep network will then be to take these task-independent word embeddings as input and convert them into embeddings that would work best for AZP resolution via supervised learning. We call our approach an *embedding matching* approach because the underlying deep network attempts to compare the embedding learned for an AZP with the embedding learned for each of its antecedents.

To our knowledge, this is the first approach to AZP resolution based on deep networks. When evaluated on the Chinese portion of the OntoNotes 5.0 corpus, our embedding matching approach to AZP resolution outperforms the CN15 model, achieving state-of-the-art results.

The rest of the paper is organized as follows. Section 2 overviews related work on zero pronoun resolution for Chinese and other languages. Section 3 describes our embedding matching approach, specifically the network architecture and the way we train and apply the network. We present our evaluation results in Section 4 and our conclusions in Section 5.

2 Related Work

Chinese ZP resolution. Early approaches to Chinese ZP resolution are *rule-based*. Converse (2006) applied Hobbs' algorithm (Hobbs, 1978) to resolve the ZPs in the CTB documents. Yeh and Chen (2007) hand-engineered a set of rules for ZP resolution based on Centering Theory (Grosz et al., 1995).

In contrast, virtually all recent approaches to this task are *learning-based*. Zhao and Ng (2007) are the first to employ a supervised learning approach to Chinese ZP resolution. They trained an AZP resolver by employing syntactic and positional features in combination with a decision tree learner. Unlike Zhao and Ng, Kong and Zhou (2010) employed context-sensitive convolution tree kernels (Zhou et al., 2008) in their resolver to model syntactic information. Chen and Ng (2013) extended Zhao and Ng's feature set with novel features that encode the context surrounding a ZP and its candidate antecedents, and exploited the coreference links between ZPs as bridges to

[1] Note that deep neural networks do *not* necessarily have to be trained in a supervised manner. In fact, in early research on extending semantic modeling using auto-encoders (Salakhutdinov and Hinton, 2007), the networks were trained in an unsupervised manner, where the model parameters were optimized for the reconstruction of the input vectors.

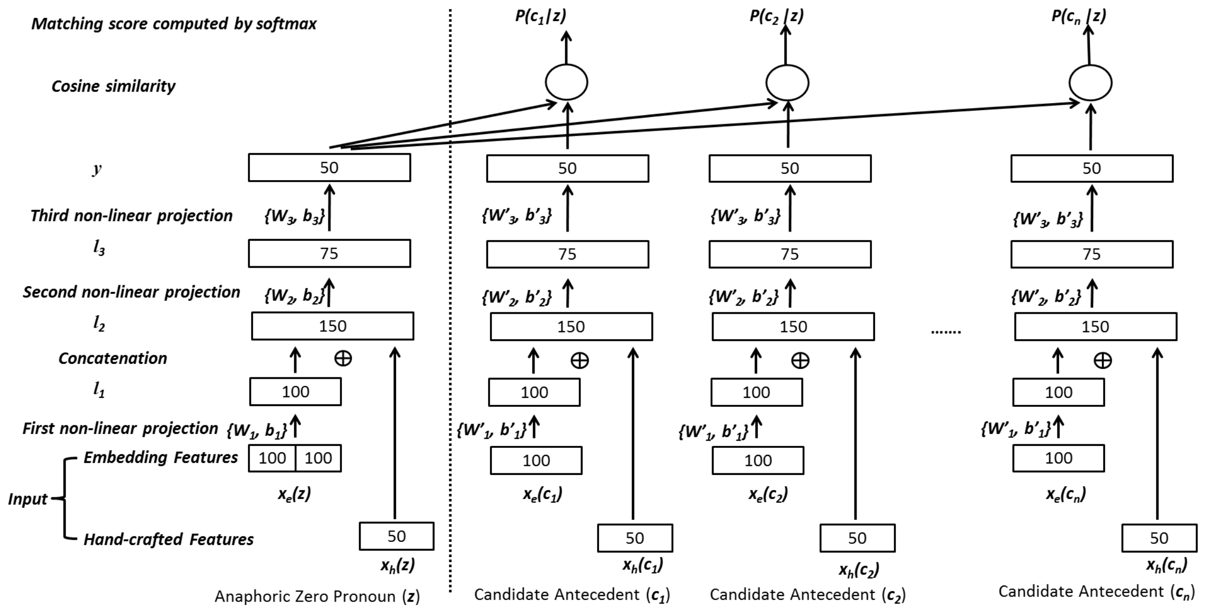

Figure 1: The architecture of our embedding matching model. The number in each box indicates the size of the corresponding vector.

find textually distant antecedents for ZPs. As mentioned above, there have been attempts to perform unsupervised AZP resolution. For instance, using only data containing manually resolved *overt* pronouns, Chen and Ng (2014a) trained a supervised overt pronoun resolver and applied it to resolve AZPs. More recently, Chen and Ng (2014b; 2015) have proposed unsupervised probabilistic AZP resolution models that rivaled their supervised counterparts in performance. While we aim to resolve *anaphoric* ZPs, Rao et al. (2015) resolved *deictic non-anaphoric* ZPs, which "refer to salient entities in the environment such as the speaker, hearer or pragmatically accessible referent without requiring any introduction in the preceding text".

ZP resolution for other languages. There have been rule-based and supervised machine learning approaches for resolving ZPs in other languages. For example, to resolve ZPs in Spanish texts, Ferrández and Peral (2000) proposed a set of hand-crafted rules that encode preferences for candidate antecedents. In addition, supervised approaches have been extensively employed to resolve ZPs in Korean (e.g., Han (2006)), Japanese (e.g., Seki et al. (2002), Isozaki and Hirao (2003), Iida et al. (2006; 2007), Sasano et al. (2008), Taira et al. (2008), Imamura et al. (2009), Sasano et al. (2009), Watanabe et al. (2010), Hayashibe et al. (2011), Iida and Poesio (2011), Sasano and Kurohashi (2011), Yoshikawa et al. (2011), Hangyo et al. (2013), Yoshino et al. (2013), Iida et al. (2015)), and Italian (e.g., Iida and Poesio (2011)).

3 Model

In this section, we first introduce our network architecture (Section 3.1), and then describe how we train it (Section 3.2) and apply it (Section 3.3).

3.1 Network Architecture

The network architecture is shown in Figure 1. Since we employ a ranking model to rank the candidate antecedents of an AZP z, the inputs to the network are (1) a feature vector representing the AZP, and (2) n feature vectors representing its n candidate antecedents, c_1, c_2, …, c_n. As will be explained in detail in Section 3.2.2, the features in each feature vector can be divided into two types: word embedding features and hand-crafted features. Each input feature vector will then be passed through three hidden layers in the network, which will successively map it into a low-dimensional feature space. The resulting vector can be viewed as the low-dimensional semantic embedding of the corresponding input vector. Finally, the model computes a *matching* score between z and each of its candidate antecedents based on their low-dimensional representations. These scores are then normalized into probabilities using a softmax.

More formally, let $x_e(z)$ and $x_h(z)$ be the vectors of embedding and hand-crafted features representing AZP z respectively, and let $x_e(c_i)$ and $x_h(c_i)$ be the vectors of embedding and hand-crafted features representing candidate antecedent c_i respectively. In addition, let $y(z)$ and $y(c_i)$ be the (low-dimensional) output vectors for z and c_i respectively, l_1, l_2, and l_3 be the intermediate hidden layers, W_i and W'_i be the weight matrices associated with z and the c_i's in hidden layer i, b_i and b'_i be the bias terms associated with z and the c_i's.[2] We then have:

$$\begin{aligned} l_1(z) &= f(W_1 x_e(z) + b_1) \\ l_2(z) &= l_1(z) \oplus x_h(z) \\ l_3(z) &= f(W_2 l_2(z) + b_2) \\ y(z) &= f(W_3 l_3(z) + b_3) \end{aligned} \tag{1}$$

$$\begin{aligned} l_1(c_i) &= f(W'_1 x_e(c_i) + b'_1) \\ l_2(c_i) &= l_1(c_i) \oplus x_h(c_i) \\ l_3(c_i) &= f(W'_2 l_2(c_i) + b'_2) \\ y(c_i) &= f(W'_3 l_3(z) + b'_3) \end{aligned} \tag{2}$$

where f is the activation function at output layer y and hidden layers l_1 and l_3. In this network, we employ tanh as the activation function. Hence,

$$f(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3}$$

The matching score between an AZP z and a candidate antecedent c_i is then measured as:

$$R(z, c_i) = \cos(y(z), y(c_i)) = \frac{y(z)^T y(c_i)}{||y(z)||||y(c_i)||} \tag{4}$$

3.2 Training

3.2.1 Training Instance Creation

We create one training instance from each AZP in each training document. Since our model is ranking-based, each training instance corresponds to an AZP z and all of its candidate antecedents C_i. In principle, we can follow previous work and assume that the set of candidate antecedents C contains all and only those maximal or modifier noun phrases (NPs) that precede z in the associated text and are at most two sentences away from it. However, to improve *training* efficiency, we select exactly four candidate antecedents for each AZP z as follows. First, we take the closest correct antecedent z to be one of the four candidate antecedents. Next, we compute a salience score for each of its *non-coreferent* candidate antecedents and select the three with the highest salience scores as the remaining three candidate antecedents.

We compute salience as follows. For each AZP z, we compute the salience score for each (partial) entity preceding z.[3] To reduce the size of the list of preceding entities, we only consider a partial entity *active* if at least one of its mentions appears within two sentences of the active AZP z. We compute the salience score of each active entity w.r.t. z using the following equation:

$$\sum_{m \in E} g(m) * decay(m) \tag{5}$$

where m is a mention belonging to active entity E, $g(m)$ is a grammatical score which is set to 4, 2, or 1 depending on whether m's grammatical role is SUBJECT, OBJECT, or OTHER respectively, and $decay(m)$ is a decay factor that is set to 0.5^{dis} (where dis is the sentence distance between m and z).

Finally, we assign the correct label (i.e., the *matching score*) to each candidate antecedent. The score is 1 for the correct antecedent and 0 otherwise.

3.2.2 Features

As we can see from Figure 1, each input feature vector, regardless of whether it is representing an AZP or one of its candidate antecedents, is composed of two types of features, embedding features and hand-crafted features, as described below.

Embedding features. To encode the lexical contexts of the AZP and its candidate antecedents, one could employ one-hot vectors. However, the resulting lexical features may suffer from sparsity. To see the reason, assuming that the vocabulary size is V and the number of neurons in the first hidden layer l_1 is L_1, the size of the weight matrices W_1 and W'_1 is $V * L_1$, which in our dataset is around two million while the number of training examples is much smaller.

Therefore, instead of using one-hot vectors, we employ embedding features. Specifically, we employ the pre-trained word embeddings (of size 100)

[2] Note that the target AZP and its candidate antecedents use different weight matrices and biases within each layer. This is needed because the features of the AZP and those of the candidate antecedents come from two different feature spaces.

[3] We compute the list of preceding entities automatically using SinoCoreferencer (Chen and Ng, 2014c), a Chinese entity coreference resolver downloadable from `http://www.hlt.utdallas.edu/~yzcchen/coreference/`.

Syntactic features (13)	whether z is the first gap in an IP clause; whether z is the first gap in a subject-less IP clause, and if so, POS(w_1); whether POS(w_1) is NT; whether w_1 is a verb that appears in a NP or VP; whether P_l is a NP node; whether P_r is a VP node; the phrasal label of the parent of the node containing POS(w_1); whether V has a NP, VP or CP ancestor; whether C is a VP node; whether there is a VP node whose parent is an IP node in the path from w_1 to C.
Other features (6)	whether z is the first gap in a sentence; whether z is in the headline of the text; the type of the clause in which z appears; the grammatical role of z (SUBJECT, OBJECT, or OTHER); whether w_{-1} is a punctuation; whether w_{-1} is a comma.

Table 1: Hand-crafted features associated with an AZP. z is a zero pronoun. V is the VP node following z. w_i is the ith word to the right of z (if i is positive) or the ith word to the left of z (if i is negative). C is lowest common ancestor of w_{-1} and w_1. P_l and P_r are the child nodes of C that are the ancestors of w_{-1} and w_1 respectively.

Syntactic features (12)	whether c has an ancestor NP, and if so, whether this NP is a descendent of c's lowest ancestor IP; whether c has an ancestor VP, and if so, whether this VP is a descendent of c's lowest ancestor IP; whether c has an ancestor CP; the grammatical role of c (SUBJECT, OBJECT, or OTHER); the clause type in which c appears; whether c is an adverbial NP, a temporal NP, a pronoun or a named entity.
Distance features (4)	the sentence distance between c and z; the segment distance between c and z, where segments are separated by punctuations; whether c is the closest NP to z; whether c and z are siblings in the associated parse tree.
Other features (2)	whether c is in the headline of the text; whether c is a subject whose governing verb is lexically identical to the verb governing of z.

Table 2: Hand-crafted features associated with a candidate antecedent. z is a zero pronoun. c is a candidate antecedent of z. V is the VP node following z in the parse tree.

obtained by training *word2vec*[4] on the Chinese portion of the training data from the OntoNotes 5.0 corpus. For an AZP z, we first find the word preceding it and its governing verb, and then concatenate the embeddings of these two words to form the AZP's embedding features. (If z happens to begin a sentence, we use a special embedding to represent the word preceding it.) For a candidate antecedent, we employ the word embedding of its head word as its embedding features.

Hand-crafted features. The hand-crafted features are (low-dimensional) features that capture the syntactic, positional and other relationships between an AZP and its candidate antecedents. These features are similar to the ones employed in previous work on AZP resolution (e.g., Zhao and Ng (2007), Kong and Zhou (2010), Chen and Ng (2013)).

We split these hand-crafted features into two disjoint sets: those associated with an AZP and those associated with a candidate antecedent. If a feature is computed based on the AZP, then we regard it as a feature associated with the AZP; otherwise, we put it in the other feature set. A brief description of the hand-crafted features associated with an AZP and those associated with a candidate antecedent are shown in Table 1 and Table 2 respectively. Note that we convert each multi-valued feature into a corresponding set of binary-valued features (i.e., if a feature has N different values, we will create N binary indicators to represent it). To ensure that the number of hand-crafted features representing an AZP is equal to the number of hand-crafted features representing a candidate antecedent[5], we append to the end of a feature vector as many dummy zeroes as needed.[6]

3.2.3 Parameter Estimation

We employ online learning to train the network, with one training example in a mini-batch. In other words, we update the weights after processing each training example based on the *correct* matching scores of the training example (which is 1 for the correct antecedent and 0 otherwise) and the network's *predicted* matching scores.

To compute the predicted matching score between AZP z and one of its candidate antecedents c_i, we apply the following softmax function:

$$P(c_i|z,\Lambda) = \frac{exp(\gamma R(z,c_i))}{\sum_{c' \in C} exp(\gamma R(z,c'))} \qquad (6)$$

where (1) γ is a smoothing factor that is empirically set on a held-out data set, (2) $R(z, c_i)$ is the cosine similarity between vector $y(z)$ and vector $y(c_i)$ (see Section 3.1), (3) $\mathcal{C}$ denotes the set of candidate antecedents of z, and (4) Λ denotes the set of parameters of our neural network:

[4] `https://code.google.com/p/word2vec/`

[5] As seen in Figure 1, we set the length of the vector to 50.

[6] Appending dummy 0s is solely for the convenience of the network implementation: doing so does not have any effect on any computation.

$$\Lambda = \{W_1, W_2, W_3, b_1, b_2, b_3, \\ W_1', W_2', W_3', b_1', b_2', b_3'\} \quad (7)$$

To maximize the matching score of the correct antecedent, we estimate the model parameters to minimize the following loss function:

$$\mathcal{J}_z(\Lambda) = -\sum_{c_i \in \mathcal{C}} \delta(z, c_i) P(c_i|z, \Lambda) \quad (8)$$

where $\delta(z, c_i)$ is an indicator function indicating whether AZP z and candidate antecedent c_i are coreferent:

$$\delta(z, c_i) = \begin{cases} 1, & \text{if } z \text{ and } c_i \text{ are coreferent} \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

Since $\mathcal{J}_z(\Lambda)$ is differentiable w.r.t. to Λ, we train the model using stochastic gradient descent. Specifically, the model parameters Λ are updated according to the following update rule:

$$\Lambda_t = \Lambda_{t-1} - \alpha \frac{\partial \mathcal{J}(\Lambda_{t-1})}{\partial \Lambda_{t-1}} \quad (10)$$

where α is the learning rate, and Λ_t and Λ_{t-1} are model parameters at the tth iteration and the $(t-1)$th iteration respectively. To avoid overfitting, we determine the hyperparameters of the network using a held-out development set.

3.3 Inference

After training, we can apply the resulting network to find an antecedent for each AZP. Each test instance corresponds to an AZP z and four of its candidate antecedents. Specifically, the four candidate antecedents with the highest salience scores will be chosen. Importantly, unlike in training, where we guarantee that the correct antecedents is among the set of candidate antecedents, in testing, we *don't*. We use the network to rank the candidate antecedents by computing the posterior probability of each of them being a correct antecedent of z, and select the one with the highest probability to be its antecedent.

The aforementioned resolution procedure can be improved, however. The improvement is motivated by a problem we observed previously (Chen and Ng, 2013): an AZP and its closest antecedent can sometimes be far away from each other, thus making it difficult to correctly resolve the AZP. To address this problem, we employ the following resolution procedure in our experiments. Given a test document, we process its AZPs in a left-to-right manner. As soon as we resolve an AZP to a preceding NP c, we fill the corresponding AZP's gap with c. Hence, when we process an AZP z, all of its preceding AZPs in the associated text have been resolved, with their gaps filled by the NPs they are resolved to. To resolve z, we create test instances between z and its four most salient candidate antecedents in the same way as described before. The only difference is that the set of candidate antecedents of z may now include those NPs that are used to fill the gaps of the AZPs resolved so far. Some of these additional candidate antecedents are closer to z than the original candidate antecedents, thereby facilitating the resolution of z. If the model resolves z to the additional candidate antecedent that fills the gap left behind by, say, AZP z', we postprocess the output by resolving z to the NP that z' is resolved to.[7]

	Training	Test
Documents	1,391	172
Sentences	36,487	6,083
Words	756,063	110,034
AZPs	12,111	1,713

Table 3: Statistics on the training and test sets.

4 Evaluation

4.1 Experimental Setup

Datasets. We employ the Chinese portion of the OntoNotes 5.0 corpus that was used in the official CoNLL-2012 shared task (Pradhan et al., 2012). In the CoNLL-2012 data, the training set and the development set contain ZP coreference annotations, but the test set does not. Therefore, we train our models on the training set and perform evaluation on the development set. Statistics on the datasets are shown in Table 3. The documents in these datasets come from six sources, namely Broadcast News (BN), Newswire (NW), Broadcast Conversation (BC), Telephone Conversation (TC), Web Blog (WB) and Magazine (MZ).

Evaluation measures. Following previous work on AZP resolution (e.g., Zhao and Ng (2007), Chen and Ng (2013)), we express the results of AZP resolution in terms of recall (R), precision (P) and F-score (F). We report the scores for each source in addition to the overall score.

[7] This postprocessing step is needed because the additional candidate antecedents are only gap fillers.

Number of embedding features for a word	100
Number of hand-crafted features	50
Number of neurons in l_1	100
Number of neurons in l_3	75
Number of neurons in y	50
Number of epochs over the training data	100
Smoothing factor γ	20
Learning rate α	0.01

Table 4: Hyperparameter values.

Hyperparameter tuning. We reserve 20% of the training set for tuning hyperparameters. The tuned hyperparameter values are shown in Table 4.

Evaluation settings. Following Chen and Ng (2013), we evaluate our model in three settings. In Setting 1, we assume the availability of gold syntactic parse trees and gold AZPs. In Setting 2, we employ gold syntactic parse trees and system (i.e., automatically identified) AZPs. Finally, in Setting 3, we employ system syntactic parse trees and system AZPs. The gold and system syntactic parse trees, as well as the gold AZPs, are obtained from the CoNLL-2012 shared task dataset, while the system AZPs are identified by a learning-based AZP identifier described in the Appendix.

Baseline system. As our baseline, we employ Chen and Ng's (2015) system, which has achieved the best result on our test set.

4.2 Results and Discussion

Results of the baseline system and our model on entire test set are shown in row 1 of Table 5. The three major columns in the table show the results obtained in the three settings. As we can see, our model outperforms the baseline significantly by 2.0%, 1.8%, and 1.1% in F-score under Settings 1, 2, and 3, respectively.[8]

Rows 2−7 of Table 5 show the resolution results on each of the six sources. As we can see, in Setting 1, our model beats the baseline on all six sources in F-score: by 2.4% (NW), 2.5% (MZ), 4.5% (WB), 1.6% (BN), 1.4% (BC), and 0.4% (TC). All the improvements are significant except for TC. These results suggest that our approach works well across different sources. In Setting 2, our model outperforms the baseline on all sources except NW and BC, where the F-scores drop insignificantly by 0.1% for both sources. Finally, in Setting 3, our model outperforms the baseline on all sources except NW and TC, where F-scores drop significantly by 0.7% for NW and 1.1% for TC.

Given the challenges in applying supervised learning (in particular, the difficulty and time involved in training the deep neural network as well as the time and effort involved in manually annotating the data needed to train the network), one may wonder whether the small though statistically significant improvements in these results provide sufficient justification for going back to supervised learning from the previous state-of-the-art unsupervised model. We believe that this is the beginning, not the end, of applying deep neural networks for AZP resolution. In particular, there is a lot of room for improvements, which may involve incorporating more sophisticated features and improving the design of the network (e.g., the dimensionality of the intermediate representations, the number of hidden layers, the objective function), for instance.

4.3 Ablation Results

Recall that the input of our model is composed of two groups of features, embedding features and hand-crafted features. To investigate the contribution of each of these two feature groups, we conduct ablation experiments. Specifically, in each ablation experiment, we retrain the network using only one group of features.

Ablation results under the three settings are shown in Table 6. In Setting 1, when the hand-crafted features are ablated, F-score drops significantly by 12.2%. We attribute the drop to the fact that the syntactic, positional, and other relationships encoded in the hand-crafted features play an important role in resolving AZPs. When the embedding features are ablated, F-score drops significantly by 3.7%. This result suggest the effectiveness of the embedding features.

Similar trends can be observed w.r.t. the other two settings: in Setting 2, F-score drops significantly by 6.8% and 2.2% when the hand-crafted features and the embedding features are ablated respectively, while in Setting 3, F-score drops significantly by 4.6% and 1.1% when the hand-crafted features and the embedding features are ablated.

4.4 Learning Curve

We show in Figure 2 the learning curve of the our model obtained under Setting 1. As we can see, after the first epoch, the F-score on the entire test set is around 46%, and it gradually increases to

[8] All significance tests are paired t-tests, with $p < 0.05$.

	Setting 1: Gold Parses, Gold AZPs						Setting 2: Gold Parses, System AZPs						Setting 3: System Parses, System AZPs					
	Baseline			Our Model			Baseline			Our Model			Baseline			Our Model		
Source	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Overall	50.0	50.4	50.2	51.8	52.5	52.2	35.7	26.2	30.3	39.6	27.0	32.1	19.6	15.5	17.3	21.9	15.8	18.4
NW	46.4	46.4	46.4	48.8	48.8	48.8	32.1	28.1	30.0	34.5	26.4	29.9	11.9	14.3	13.0	11.9	12.8	12.3
MZ	38.9	39.1	39.0	41.4	41.6	41.5	29.6	19.6	23.6	34.0	22.4	27.0	4.9	4.7	4.8	9.3	7.3	8.2
WB	51.8	51.8	51.8	56.3	56.3	56.3	39.1	22.9	28.9	44.7	25.1	32.2	20.1	14.3	16.7	23.9	16.1	19.2
BN	53.8	53.8	53.8	55.4	55.4	55.4	30.8	30.7	30.7	36.9	31.9	34.2	18.2	22.3	20.0	22.1	23.2	22.6
BC	49.2	49.6	49.4	50.4	51.3	50.8	35.9	26.6	30.6	37.6	25.6	30.5	19.4	14.6	16.7	21.2	14.6	17.3
TC	51.9	53.5	52.7	51.9	54.2	53.1	43.5	28.7	34.6	46.3	29.0	35.6	31.8	17.0	22.2	31.4	15.9	21.1

Table 5: AZP resolution results of the baseline and our model on the test set.

	Setting 1: Gold Parses Gold AZPs			Setting 2: Gold Parses System AZPs			Setting 3: System Parses System AZPs		
System	R	P	F	R	P	F	R	P	F
Full system	51.8	52.5	52.2	39.6	27.0	32.1	21.9	15.8	18.4
Embedding features only	39.2	40.8	40.0	30.9	21.5	25.3	16.3	12.0	13.8
Hand-crafted features only	48.2	48.7	48.5	37.0	25.1	29.9	20.6	14.9	17.3

Table 6: Ablation results of AZP resolution on the whole test set.

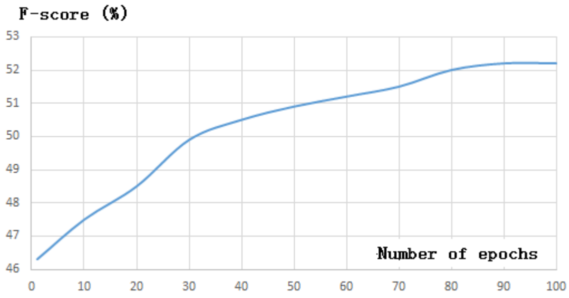

Figure 2: The learning curve of our model on the entire test set under Setting 1.

52% in the 80th epoch when performance starts to plateau. These results provide suggestive evidence for our earlier hypothesis that our objective function (Equation (8)) is tightly coupled with the desired evaluation metric (F-score).

4.5 Analysis of Results

To gain additional insights into our approach, we examine the outputs of our model obtained under Setting 1.

We first analyze the cases where the AZP was correctly resolved by our model but incorrectly resolved by the baseline. Consider the following representative example with the corresponding English translation.

[陈水扁] 在登机前发表简短谈话时表示，[台湾] 要站起来走出去。... $*pro*$ 也希望此行能把国际友谊带回来。
[Chen Shui-bian] delivered a short speech before boarding, saying that [Taiwan] should stand up and go out. ... $*pro*$ also hopes that this trip can bring back international friendship.

In this example, the correct antecedent of the AZP is 陈水扁 (Chen Shui-bian). However, the baseline incorrectly resolves it to 台湾 (Taiwan). The baseline's mistake can be attributed to the facts that (1) 台湾 is the most salient candidate antecedent in the discourse, and (2) 台湾 is closer to the AZP than the correct antecedent 陈水扁. Nevertheless, our model still correctly identifies 陈水扁 as the AZP's antecedent because of the embedding features. A closer inspection of the training data reveals that although the word 陈水扁 never appeared as the antecedent of an AZP whose governing verb is 希望 (hope) in the training data, many AZPs that are governed by 希望 are coreferent with other person names. Because the word 陈水扁 has a similar word embedding as those person names, our approach successfully generalizes such lexical context and makes the right resolution decision.

Next, we examine the errors made by our model and find that the majority of the mistakes result from insufficient lexical contexts. Currently, to encode the lexical contexts, we only consider the word preceding the AZP and its governing verb, as well as the head word of the candidate antecedent. However, this encoding ignored a lot of potentially useful context information, such as the clause following the AZP, the modifier of the candidate antecedent and the clause containing the candidate antecedent. Consider the following example:

[我] 前一会精神上太紧张。...*pro* 现在比较平静了。
[I] was too nervous a while ago. ... *pro* am now calmer.

To resolve the AZP to its correct antecedent 我 (I), one needs to compare the two clauses containing the AZP and 我. However, since our model does not encode a candidate antecedent's context, it does not resolve the AZP correctly. One way to address this problem would be to employ *sentence embeddings* to represent the clauses containing the AZP and its candidate antecedents, and then perform sentence embedding matching to resolve the AZP. The primary challenge concerns how to train the model to match two clauses with probably no overlapping words and with a limited number of training examples.

5 Conclusions

We proposed an embedding matching approach to zero pronoun resolution based on deep networks. To our knowledge, this is the first neural network-based approach to zero pronoun resolution. When evaluated on the Chinese portion of the OntoNotes corpus, our approach achieved state-of-the-art results.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 601--612.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1360--1365.

Chen Chen and Vincent Ng. 2014a. Chinese zero pronoun resolution: An unsupervised approach combining ranking and integer linear programming. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1622--1628.

Chen Chen and Vincent Ng. 2014b. Chinese zero pronoun resolution: An unsupervised probabilistic model rivaling supervised resolvers. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 763--774.

Chen Chen and Vincent Ng. 2014c. SinoCoreferencer: An end-to-end Chinese event coreference resolver. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4532--4538.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 320--326.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405--1415.

Susan Converse. 2006. *Pronominal Anaphora Resolution in Chinese*. Ph.D. thesis, University of Pennsylvania.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660--669.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971--1982.

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in Spanish. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 166--172.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203--226.

Na-Rae Han. 2006. *Korean zero pronouns: Analysis and resolution*. Ph.D. thesis, University of Pennsylvania.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 924--934.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 201--209.

Jerry Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311--338.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 804--813.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 625--632.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing*, 6(4).

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2179--2189.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85--88.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 184--191.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44--56. MIT Press.

Fang Kong and GuoDong Zhou. 2010. A tree kernel-based unified framework for Chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882--891.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. In *Transactions of the Association for Computational Linguistics*, 3:405--418.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: Shared Task*, pages 1--40.

Sudha Rao, Allyson Ettinger, Hal Daumé III, and Philip Resnik. 2015. Dialogue focus tracking for zero pronoun resolution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 494--503.

Ruslan Salakhutdinov and Geoffrey Hinton. 2007. Semantic hashing. In *Proceedings of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 758--766.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for Japanese zero anaphora resolution. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 769--776.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2009. The effect of corpus size on case frame acquisition for discourse analysis. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 521--529.

Kazuhiro Seki, Atsushi Fujii, and Tetsuya Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 523--532.

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2010. A structured model for joint learning of argument roles and predicate senses. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 98--102.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416--1426.

Yaqin Yang and Nianwen Xue. 2010. Chasing the ghost: recovering empty categories in the Chinese Treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1382--1390.

Ching-Long Yeh and Yi-Chun Chen. 2007. Zero anaphora resolution in Chinese with shallow parsing. *Journal of Chinese Language and Computing*, 17(1):41--56.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with Markov Logic. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1125--1133.

Koichiro Yoshino, Shinsuke Mori, and Tatsuya Kawahara. 2013. Predicate argument structure analysis using partially annotated corpora. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 957--961.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*, pages 541--550.

GuoDong Zhou, Fang Kong, and Qiaoming Zhu. 2008. Context-sensitive convolution tree kernel for pronoun resolution. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 25--31.

Appendix: Anaphoric Zero Pronoun Identification

Recall that Settings 2 and 3 in our evaluation involve the use of system AZPs. Our supervised AZP identification procedure is composed of two steps. First, in the *extraction* step, we heuristically extract ZPs. Then, in the *classification* step, we train a classifier to determine which of the ZPs extracted in the first step are AZPs.

To implement the extraction step, we use Zhao and Ng's (2007) observation: ZPs can only occur before a VP node in a syntactic parse tree. However, according to Kong and Zhou (2010), ZPs do not need to be extracted from every VP: if a VP node occurs in a coordinate structure or is modified by an adverbial node, then only its parent VP node needs to be considered. We extract ZPs from all VPs that satisfy the above constraints.

To implement the classification step, we train a binary classifier using SVM^{light} (Joachims, 1999) on the CoNLL-2012 training set to distinguish AZPs from non-AZPs. Each instance corresponds to a ZP extracted in the first step and is represented by 32 features, 13 of which were proposed by Zhao and Ng (2007) and 19 of which were proposed by Yang and Xue (2010). A brief description of these features can be found in Table 7.

Syntactic features (13)	whether z is the first gap in an IP clause; whether z is the first gap in a subject-less IP clause, and if so, POS(w_1); whether POS(w_1) is NT; whether t_1 is a verb that appears in a NP or VP; whether P_l is a NP, QP, IP or ICP node; whether P_r is a VP node; the phrasal label of the parent of the node containing POS(t_1); whether V has a NP, VP, QP or CP ancestor; whether C is a VP node; whether the parent of V is an IP node; whether V's lowest IP ancestor has (1) a VP node as its parent and (2) a VV node as its left sibling; whether there is a VP node whose parent is an IP node in the path from t_1 to C.
Lexical features (13)	the words surrounding z and/or their POS tags, including w_1, w_{-1}, POS(w_1), POS(w_{-1}) + POS(w_1), POS(w_1) + POS(w_2), POS(w_{-2}) + POS(w_{-1}), POS(w_1) + POS(w_2) + POS(w_3), POS(w_{-1}) + w_1, and w_{-1} + POS(w_1); whether w_1 is a transitive verb, an intransitive verb or a preposition; whether w_{-1} is a transitive verb without an object.
Other features (6)	whether z is the first gap in a sentence; whether z is in the headline of the text; the type of the clause in which z appears; the grammatical role of z; whether w_{-1} is a punctuation; whether w_{-1} is a comma.

Table 7: Features for AZP identification. z is a zero pronoun. V is the VP node following z. w_i is the ith word to the right of z (if i is positive) or the ith word to the left of z (if i is negative). C is lowest common ancestor of w_{-1} and w_1. P_l and P_r are the child nodes of C that are the ancestors of w_{-1} and w_1 respectively.

When gold parse trees are employed, the recall, precision and F-score of the AZP identifier on our test set are 75.1%, 50.1% and 60.1% respectively. Using automatic parse trees, the performance of the AZP identifier drops to 43.7% (R), 30.7% (P) and 36.1% (F).