

Joint Coreference Resolution for Zeros and non-Zeros in Arabic

Abdulrahman Aloraini^{1,2} Sameer Pradhan^{3,4} Massimo Poesio¹

¹School of Electronic Engineering and Computer Science, Queen Mary University of London

²Department of Information Technology, College of Computer, Qassim University

³cemantix.org

⁴Linguistic Data Consortium, University of Pennsylvania, Philadelphia, USA

a.aloraini@qmul.ac.uk spradhan@cemantix.org m.poesio@qmul.ac.uk
upenn.edu

Abstract

Most existing proposals about anaphoric zero pronoun (AZP) resolution regard full mention coreference and AZP resolution as two independent tasks, even though the two tasks are clearly related. The main issues that need tackling to develop a joint model for zero and non-zero mentions are the difference between the two types of arguments (zero pronouns, being null, provide no nominal information) and the lack of annotated datasets of a suitable size in which both types of arguments are annotated for languages other than Chinese and Japanese. In this paper, we introduce two architectures for jointly resolving AZPs and non-AZPs, and evaluate them on Arabic, a language for which, as far as we know, there has been no prior work on joint resolution. Doing this also required creating a new version of the Arabic subset of the standard coreference resolution dataset used for the CoNLL-2012 shared task (Pradhan et al., 2012) in which both zeros and non-zeros are included in a single dataset.

1 Introduction

In pronoun-dropping (pro-drop) languages such as Arabic (Eid, 1983), Chinese (Li and Thompson, 1979), Italian (Di Eugenio, 1990) and other romance languages (e.g., Portuguese, Spanish), Japanese (Kameyama, 1985), and others (Young-Joo, 2000), arguments in syntactic positions in which a pronoun is used in English can be omitted. Such arguments—sometimes called null arguments, empty arguments, or zeros, and called anaphoric zero pronouns (AZP) here when they are anaphoric, are illustrated by the following example:

... المفارقة الأخرى عن **بوش** هي عدم حماسته للمؤتمر الدولي
، لأن **بوش** من البداية ، يريد * اجتماعا مختلفا ...

*Ironically, **Bush** did not show any enthusiasm for the international conference, because **Bush** since the beginning, (he) wanted to attend another conference ...*

In the example, the ’*’ is an anaphoric zero pronoun—a gap replacing an omitted pronoun which refers to a previously mentioned noun, i.e. **Bush**.¹

Although AZPs are common in pro-drop languages (Chen and Ng, 2016), they are typically not considered in standard coreference resolution architectures. Existing coreference resolution systems for Arabic would cluster the overt mentions of **Bush**, but not the AZP position; vice versa, AZP resolution systems would resolve the AZP, to one of the previous mentions, but not other mentions. The main reason for this is that AZPs are empty mentions, meaning that it is not possible to encode features commonly used in coreference systems—the head, syntactic and lexical features as in pre-neural systems. As a result, papers such as (Iida et al., 2015) have shown that treating the resolution of AZPs and realized mentions separately is beneficial. However, it has been shown that the more recent language models and end-to-end systems do not suffer from these issues to the same extent. BERT, for example, learns surface, semantic and syntactic features of the whole context (Jawahar et al., 2019) and it has been shown that BERT encodes sufficient information about AZPs within its layers to achieve reasonable performance (Aloraini and Poesio, 2020b,a). However, these findings have not yet led to many coreference resolution models attempting to resolve both types of mentions in a single learning framework (in fact, we are only aware of two, (Chen et al., 2021; Yang et al., 2022), the second of which was just proposed) and these have not been evaluated with Arabic.

In this paper, we discuss two methods for jointly clustering AZPs and non-AZPs, that we evaluate on Arabic: a *pipeline* and a *joint learning* architecture. In order to train and test these two architectures, however, it was also necessary to create a

¹We use here the notation for AZPs used in the Arabic portion of OntoNotes 5.0, in which AZPs are denoted as * and we also use another notation which is ***pro***.

new version of the Arabic portion of the CoNLL-2012 shared task corpus in which both zeros and non-zeros are annotated in the same documents. To summarize, the contributions of this paper are as follows:

- We introduce two new architectures for resolving AZPs and non-AZPs together, the *pipeline* and the *joint learning* architecture. One of our architectures, the *joint learning*, outperforms the one existing joint end-to-end model (Chen et al., 2021) when resolving both types of mentions together.
- We create an extended version of the Arabic portion of CoNLL-2012 shared task in which the zero and non-zero mentions are represented in the same document. The extended dataset is suitable for training AZPs and non-AZPs jointly or each type separately.

2 Related Work

Most existing works regard coreference resolution and AZP resolution as two independent tasks. Many studies were dedicated to Arabic coreference resolution using CoNLL-2012 dataset (Li, 2012; Zhekova and Kübler, 2010; Björkelund and Nugues, 2011; Stamborg et al., 2012; Uryupina et al., 2012; Fernandes et al., 2014; Björkelund and Kuhn, 2014; Aloraini et al., 2020; Min, 2021), but AZPs were excluded from the dataset so no work considered them. Aloraini and Poesio (2020b) proposed a BERT-base approach to resolve AZPs to their true antecedent, but they did not resolve other mentions.

There have been a few proposals on solving the two tasks jointly for other languages. Iida and Poesio (2011) integrated the AZP resolver with a coreference resolution system using an integer-linear-programming model. Kong and Ng (2013) employed AZPs to improve the coreference resolution of non-AZPs using a syntactic parser. Shibata and Kurohashi (2018) proposed an entity-based joint coreference resolution and predicate argument structure analysis for Japanese. However, these works relied on language-specific features and some assumed the presence of AZPs.

There are two end-to-end neural proposals about learning AZPs and non-AZPs together. The first proposal is by Chen et al. (2021) who combined tokens and AZP gaps representations using an encoder. The two representations interact in a two-stage mechanism to learn their coreference infor-

mation, as shown in Figure 5. The second proposal, just published, is by (Yang et al., 2022), who proposed the CorefDPR architecture. CorefDPR consists of four components: the input representation layer, coreference resolution layer, pronoun recovery layer and general CRF layer. In our experiments, we only compared our results with the first proposal because the second system was only evaluated on the Chinese conversational speech of OntoNotes² and the model is not publicly available which makes it difficult to compare our results with theirs.

3 An Extended Version of the CoNLL Arabic dataset with AZPs

The goal of the CoNLL-2012 coreference shared task is to learn coreference resolution for three languages (English, Chinese and Arabic). However, AZPs were excluded from the task even though they are annotated in OntoNotes Arabic and Chinese. This was because considering AZPs decreased the overall performance on Arabic and Chinese (Pradhan et al., 2012), but not on English because it is not a pro-drop language (White, 1985). So in order to study joint coreference resolution for explicit mentions and zero anaphors, we had to create a novel version of the CoNLL-2012 dataset in which AZPs and all related information are included. The CoNLL-2012 annotation layers consists of 13 layers and they are in Appendix A.

Existing proposals evaluated their AZP systems using OntoNotes Normal Forms (ONF)³. They are annotated with AZPs and other mentions; however, they are not as well-prepared as CoNLL-2012. To create a CoNLL-like dataset with AZPs, we extract AZPs from ONF and add them to the already-existing CoNLL files. The goal of the new dataset is to be suitable for clustering AZPs and non-AZPs, and can be compared with previous proposals that did not consider AZPs and as well as with future works that consider them.

To include AZPs and their information (e.g., Part-of-Speech and parse tree) to CoNLL-2012, we can use ONF. However, while adding AZPs to the clusters, we realized that there is one difficulty: some

²The TC part of the Chinese portion in OntoNotes.

³The OntoNotes Normal Form (ONF) was originally meant to be a human-readable integrated representation of the multiple layers in OntoNotes. However, it has been used by many as a machine readable representation—as it is also more or less true—to extract annotations, primarily zeros that are typically excluded from the traditional CoNLL tabular representation.

Chain 71	(IDENT)	6.2-13 7.2-2	الجيش الشعبي لـ -تحرير السودان " سمسون خواجه *
Chain 92	(IDENT)	8.1-11 8.16-16	وزارة الخارجية السودانية مطرف صديق الذي يزأس الحكومي ه
Chain 95	(APPOS)	ATTRIB 8.1-4 HEAD 8.5-6	وكيل وزارة الخارجية السودانية مطرف صديق

Figure 1: A screenshot of OntoNotes Normal Forms (onf). Chain 71 is not considered part of a CoNLL-2012 shared task because the cluster would become a singleton when we remove the AZP (denoted as *).

coreference chains only exist in ONF, but not in CoNLL-2012. These are clusters consisting of only one mention and one AZP, as in the example illustrated in Figure 1. Chain 71 has two mentions, an AZP (denoted with *) and a mention. Since CoNLL-2012 does not consider AZPs in coreference chains, this cluster would only have a single mention because CoNLL-2012 removed AZPs (these clusters are known as singletons, contains only one mention). Our new dataset includes AZPs; therefore, such clusters should be included. To add them to the existing CoNLL-2012, we have to assign them a new cluster. We did this by writing a script that automatically extracts AZPs from ONF and adds them in CoNLL-2012 following these steps:

1. Finds all clusters that have AZPs in ONF and extracts AZPs.
2. Each extracted AZP is either:
 - (a) Clustered with two or more mentions: For this case, CoNLL has already assigned a coreference-chain number and we assign the AZP to the same number.
 - (b) Clustered with only one mention: We create a new cluster that include the single mention and the AZP.
3. Adds the AZP and writes other relevant information, such as, Part-of-Speech, syntax, and all the annotation layers.

Adding AZPs to CoNL-2012 is beneficial to learn how to resolve them with other mentions or can be useful for future CoNLL-shared tasks and any other related NLP task. After preparing the new CoNLL dataset as discussed, we used it to train the joint coreference model. This new version

Category	Training	Development	Test
Documents	359	44	44
Sentences	7,422	950	1,003
Words	264,589	30,942	30,935
AZPs	3,495	474	412

Table 1: The documents, sentences, words and AZPs of the extended version of CoNLL-2012. We follow the same split as in the original CoNLL-2012 for training, development and test.

of Arabic OntoNotes will be made available with the next release of OntoNotes. The distribution of documents, sentences, words, and AZPs of this extended dataset are in Table 1.

4 The Models

Earlier proposals resolved AZPs based on the antecedents that are in the same sentence as the AZP or two sentences away (Chen and Ng, 2015, 2016; Yin et al., 2016, 2017; Liu et al., 2017; Yin et al., 2018; Aloraini and Poesio, 2020b). However, it has been shown that learning mention coreference in the whole document is beneficial for AZP resolution (Chen et al., 2021). Therefore, we apply two novel methods for resolving AZPs using clusters and coreference chains. The *pipeline* resolves AZPs based on the output clusters from the coreference resolution model while the *joint learning* learns how to resolve AZPs from the coreference chains, we show an example of these two in Figure 2. In the example, the *pipeline* resolves AZPs to clusters, instead of mentions and the *joint learning* finds the coreference chains for mentions, including AZPs. Earlier proposals suffered from two main problems. First, they consider a limited number of candidates (i.e mentions in two sentences away

from the AZP) as possible true antecedents; however, the true antecedent might be far away from the AZP. Second, other mentions can share salient context as the true antecedent which can introduce more noise to the learning. Our methods mitigate these problems by considering all mentions in the document and employing more relevant information. The *pipeline* resolves AZPs based on clusters which decreases dramatically the number of AZP candidates. The *joint learning* resolves AZPs using coreference chains which incorporates broader context for AZPs, insufficient contexts results in many errors (Chen and Ng, 2016).

4.1 The Pipeline Model

In a *pipeline* setting, the inputs are the extended version of CoNLL, the one we described in Section 3. Each file consists of multiple sentences and we follow the same splits in CoNLL-2012 (Pradhan et al., 2012) for train, development and test. We initially fed the documents for training into two models: *coreference resolution* and *AZP identification*. We used the Arabic coreference resolution by (Aloraini et al., 2020) and the proposed AZP identification by (Aloraini and Poesio, 2020a). The outputs of coreference resolutions are clusters and each one has its own mentions. The outputs of the AZP identification are the predicted gap positions of AZPs. The *AZP resolution* model by (Aloraini and Poesio, 2020b) learns how to resolve the identified AZPs with their clusters. We show how we represent the input in the following:

The *input* is a document with sentences separated with periods, and has a total of n words. The *input* does not consider AZPs initially, they are masked.

$$input = (w_1, w_2, w_3, \dots, w_n) \quad (1)$$

We first feed the *input* into the *coreference resolution* model which outputs the mention clusters, c_1, c_2 , to the last cluster index, k .

$$output_clusters = coref_res(input) \quad (2)$$

$$output_clusters = (c_1, c_2, \dots, c_k) \quad (3)$$

After finding the coreference clusters, the *AZP Identification* model predicts the AZP positions in two steps. First, the *AZP identification* uses a Part-of-Speech tool to tag words and mark gaps after verbs as potential AZPs. Second, *AZP identification* classifies these marked gaps as AZPs or not.

Therefore, not every gap between words has an AZP. For example, in (5) there is no AZP between the words w_2 and w_3 , but there is one between w_1 and w_2 (i.e. a_i). We find AZP locations and extract their positions.

$$input_with_azp = AZP_Id(input) \quad (4)$$

$$input_with_azp = (w_1, a_i, w_2, w_3, \dots, w_n) \quad (5)$$

$$AZPs = (a_i, \dots, a_k) \quad (6)$$

$$ss = same_sentence(a_i, c_j) \quad (7)$$

$$cd = azp_cluster_distance(a_i, c_j) \quad (8)$$

$$AZP_i = (a_i_pre, a_i_next, ss, cd) \quad (9)$$

We follow the same representation for AZPs as (Aloraini and Poesio, 2020b):

- embeddings for previous word to AZP.
- embeddings for next word to AZP.
- Whether the AZP and the candidate entity (represented either as the last mention or first mention) are in the same sentence or not.
- The distance between the AZP and its cluster representation.

The four features are concatenated, as shown in (9).

Clusters can be represented in different ways, including, e.g, the representation of the first mention or the last mention. We found empirically that representing clusters with the nearest mention to the AZP (the last added mention to the cluster) produces better results.

$$c_i = \{m_1, m_2, \dots, m_z\} \quad (10)$$

$$c_i = \begin{cases} m_1 & \text{the first mention to represent } c_i \\ m_z & \text{the last mention to represent } c_i \end{cases} \quad (11)$$

Next, the AZP and cluster representations are joined together through a concatenation layer. The variable *input* contains the concatenated representation of a mention pair - the AZP and its corresponding cluster. The binary variable *AZP res* receives *input* and is 1 if the AZP and the cluster corefer. The model also outputs the final clusters.

The following equations specify how the output of the network is computed:

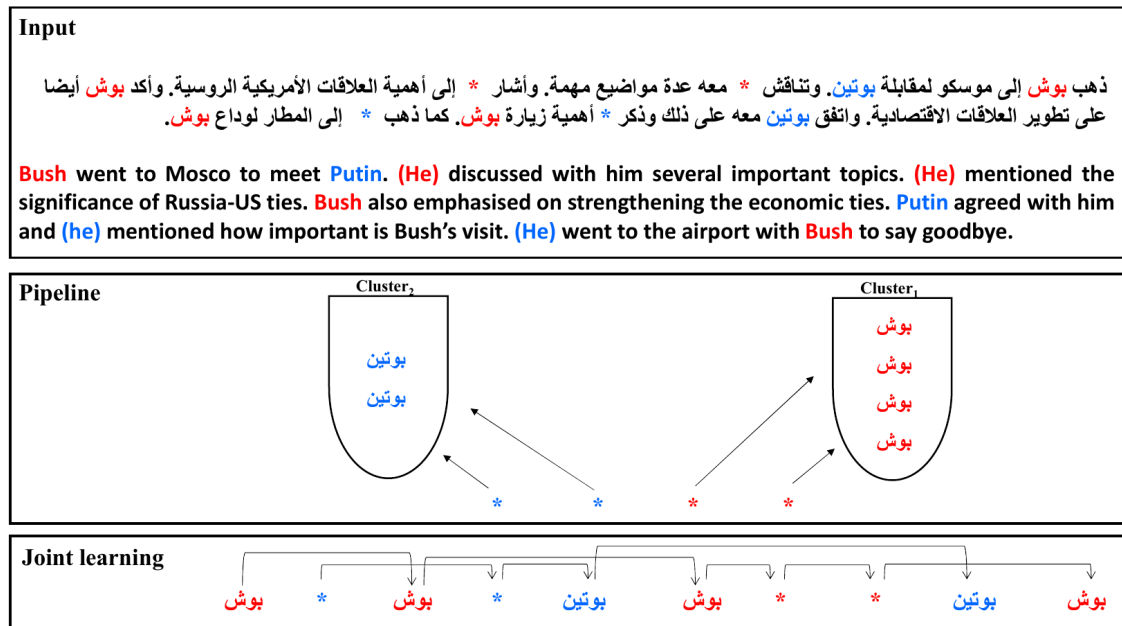


Figure 2: The input is a document and the asterisk * represents the AZPs in the text. For AZP resolution, The pipeline resolves AZPs with the output clusters and the joint learning resolves AZPs based on coreference chains.

$$input = concat(c_i, a_j) \quad (12)$$

$$input = [c_i, a_j] \quad (13)$$

$$results = AZP_Res(input) \quad (14)$$

$$results = (r_1, r_2, \dots, r_s) \quad (15)$$

The variable *results* consists of the final clusters of the resolved AZPs and non-AZPs. We show the model architecture in Figure 3.

4.2 The Joint Learning Model

Our *joint learning* architecture learns to resolve AZPs by using the explicitly represented AZP gaps. This way, AZPs would be learned as any other overt mention. In our extended CoNLL-2012 documents, AZPs have the special identified *pro*. Table 2 shows an example of a CoNLL-2012 original sentence and its extended version. However, we consider AZPs only in the training phase when we apply the coreference resolution model. At test time, AZPs are not considered, same as in a real life application. Instead, we use the *AZP identification* model by (Aloraini and Poesio, 2020b) to tag AZP gaps. After tagging, the input is ready for clustering using the trained coreference resolution model. This is how we represent the inputs for both training and testing:

The *input* is a CoNLL-2012 document with many sentences that has a set of n mentions. A mention

can be a word or an AZP tag (*pro*).

$$input = (m_1, m_2, m_3, \dots, m_n) \quad (16)$$

The variable *input* is fed into the *coreference resolution* (*coref_res*) model which outputs clusters. The clusters contain mentions and AZPs that refer to the same entity.

$$output_clusters = coref_res(input) \quad (17)$$

$$output_clusters = (c_1, c_2, \dots, c_k) \quad (18)$$

For the test phase, we assume a document is not labeled with AZP tags, which reflects real-life applications. Therefore, we first feed *input* into the *AZP identification* (*AZP_Id*) which outputs *input_with_azp*, that is *input* but with tagged AZPs. The *AZP identification* is pre-trained on the train set of CoNLL-2012 to detect AZP locations.

$$input_with_azp = AZP_Id(input) \quad (19)$$

$$input_with_azp = (w_1, a_i, w_2, \dots, m_n) \quad (20)$$

After preparing *input_with_azp*, we feed it into the trained *coreference resolution* model which outputs the clusters.

$$results = coref_res(input_with_azp) \quad (21)$$

$$results = (r_1, r_2, \dots, r_s) \quad (22)$$

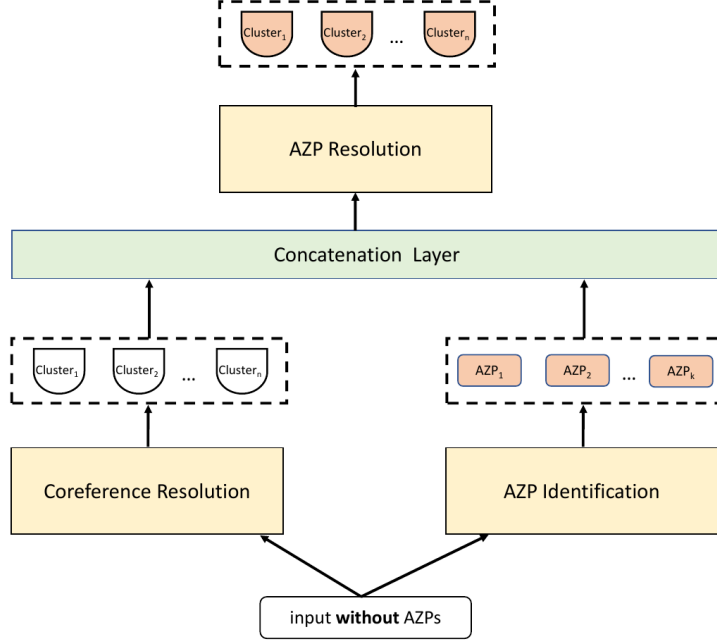


Figure 3: The input without AZPs is fed into the *Coreference Resolution* and *AZP identification* models. The outputs of the two models are clusters and AZPs respectively. Their representations are concatenated, and then their coreference information is learned through the *AZP Resolution* model.

Original CoNLL-2012 sentence	كانا في الوضع نفسه
Extended CoNLL-2012 sentence	كانا *pro* في الوضع نفسه

Table 2: An example of how we explicitly represent AZPs.

The variable *results* has the resolved AZPs and non-AZPs. We show the overall architecture in Figure 4.

5 Evaluation metrics

5.1 Coreference resolution

For our evaluation of the coreference system, we use the official CoNLL-2012 evaluation metrics to score the predicted clusters. We report recall, precision, and F_1 scores for MUC, B³ and CEAF _{ϕ_4} and the average F_1 score of those three metrics.

5.2 AZP resolution

We evaluate AZP resolution in terms of recall and precision, as defined in (Zhao and Ng, 2007):

$$\text{Recall} = \frac{\text{AZPhits}}{\text{Number of AZPs in Key}}$$

$$\text{Precision} = \frac{\text{AZPhits}}{\text{Number of AZPs in Response}}$$

Key represents the gold set of AZP entities in the dataset, and *Response* represents the predicted resolved AZPs. *AZP hits* are the reported resolved AZP positions in *Response* which occur in the same position as in *Key*.

6 Training Objectives

6.1 Pipeline

The training objective of the AZP identification is binary cross-entropy loss, as introduced in (Aloraini and Poesio, 2020a):

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^N [y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)] \quad (23)$$

θ is the set of learning parameters in the model. N is the number of training samples in the extended CoNLL-2012. y_i is the true label i and \hat{y}_i is its predicted label.

For the AZP resolution, the goal is to minimize the cross entropy error between every AZP and its

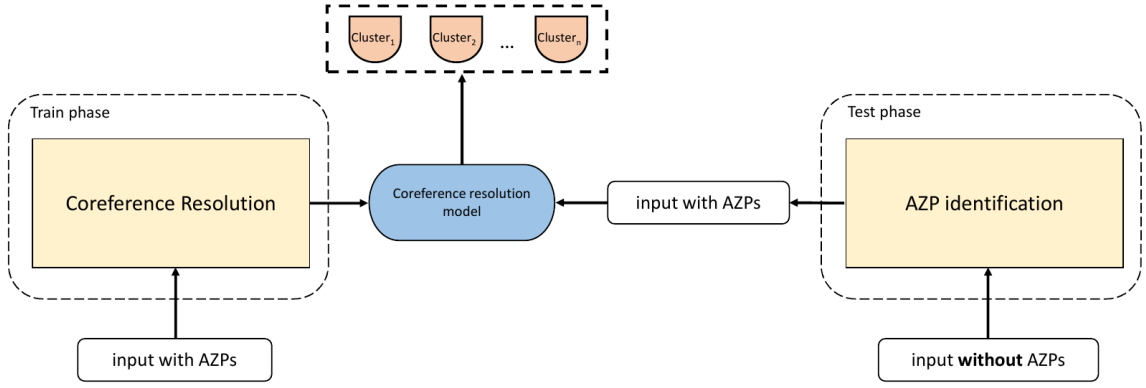


Figure 4: In the train phase, the model learns how to resolve mentions and AZPs. AZPs are represented with the *pro* tag and treated like any other mention. The test phase predicts and tags AZPs locations. We use the model proposed by (Aloraini and Poesio, 2020a) to find AZPs. The pretrained coreference resolution model is used in the test phase to cluster mentions and AZPs.

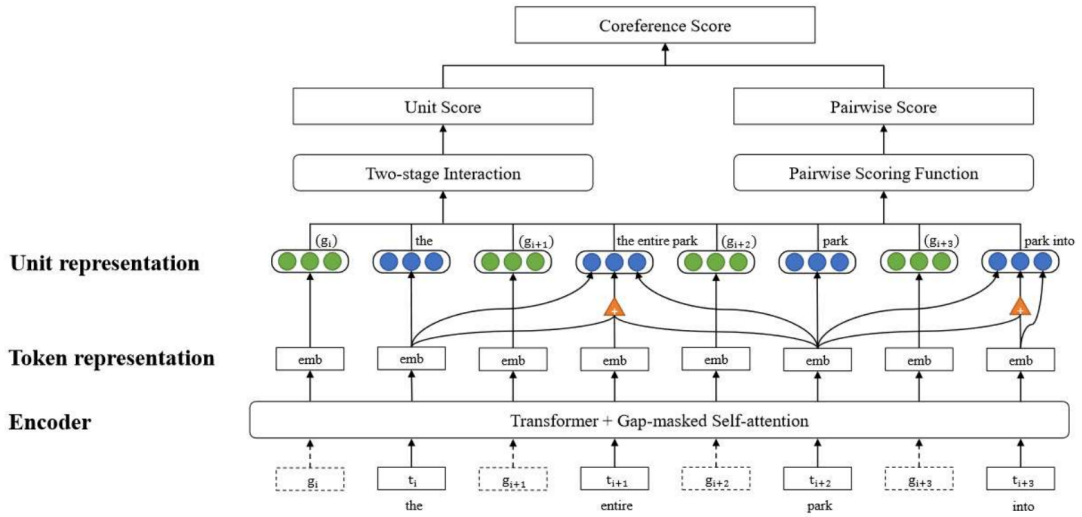


Figure 5: Resolving AZPs and non-AZPs in an end-to-end model (Chen et al., 2021).

antecedents, as defined in Aloraini and Poesio’s (2020b) model; however, we resolve AZPs with clusters, instead of mentions:

$$L(\theta) = - \sum_{t \in T} \sum_{c \in C} \delta(azp, c) \log(P(azp, c)) \quad (24)$$

T consists of the n training instances of AZPs, and C represents the k candidate clusters from the coreference resolution. $\delta(azp, c)$ returns whether a candidate cluster c is the correct one for the azp , or not. $\log(P(azp, c))$ is the predicted log probability of the (azp, c) pair.

The training objective of the coreference resolution is to optimize the log-likelihood of all correct mentions (Lee et al., 2017), as the following :

$$L(\theta) = \log \prod_{i=1}^N \sum_{\hat{y} \in \mathcal{Y}(i) \cap G(i)} P(\hat{y}) \quad (25)$$

G represents the spans in the gold cluster that includes i .

6.2 Joint Learning

In the *joint learning*, we only use the (24) for training. AZPs are treated as any other mention; therefore, they become part of the coreference resolution learning objective. We also do not have to train the AZP identification model because we only use the AZP identification in the test phase and we use the pre-trained one on the original CoNLL-2012 from (Aloraini and Poesio, 2020a).

Models	MUC			B ³			CEAF _{ϕ_4}			CoNLL Average
	R	P	F ₁	R	P	F ₁	R	P	F ₁	F ₁
Pipeline	62.9	70.7	66.5	57.3	65.6	61.2	61.1	64.5	62.7	63.5
Joint learning	65.2	75.5	70.0	62.6	68.3	65.3	64.8	67.7	66.2	67.1
Chen et al. (2021)	62.7	71.1	66.6	58.5	65.7	61.6	61.4	67.2	64.2	64.2

Table 3: Resolving AZPs and non-AZPs together.

7 Results

We compare the results of the *pipeline* and *joint learning* models with the results of Chen et al. (2021). We followed Chen et al. (2021)’s approach for hyperparameter tuning, but we changed the language model to AraBERT-base (Antoun et al., 2020). We evaluate two tasks. First, we assess the results at joint coreference resolution of both AZPs and non-AZPs. Second, we evaluate AZP resolution only. Unlike previous proposals that resolve AZPs with their antecedents, the AZPs of our methods and the Chen et al.’s (2021) model resolve AZPs differently. The *pipeline* uses the output clusters, the *joint learning* uses the coreference chains and Chen et al. (2021) uses two scoring components.

7.1 Resolving AZPs and non-AZPs

In Table 3, we see the results of resolving AZPs and non-AZPs. Chen et al.’s (2021) model achieves 64.2% F₁ score, which is 0.7% more than the *pipeline*, but less than the *joint learning* with 2.9%. Our *joint learning* approach outperforms our *pipeline* and Chen et al.’s (2021) system, achieving the best F₁ average score of 67.1%.

7.2 AZP resolution

Next, we compare the AZP resolution results. For the *pipeline*, we used two settings to represent clusters. First, we used the first mention in the cluster to be concatenated with the AZP representation. Second, we used the last-added mention. The *pipeline* approach achieves an F₁ score of 58.08% when using the first mention as the cluster representation and 58.59% when using the last mention. The *joint learning* provided better results with an F₁ score of 59.33%. Chen et al.’s (2021) model resolved more AZPs correctly than the *pipeline* and *joint learning* methods, achieving an F₁ score of 59.49% which is 0.19% more than the *joint learning* score. It seems the two components of Chen et al.’s (2021)

model, the Unit Score and Pairwise Score, are able to distinguish AZPs and mentions effectively for the AZP resolution. However, for coreference resolution, they have showed the performance is better when they did not consider AZPs as part of the coreference resolution.

Training Settings	Test Evaluation		
	P	R	F ₁
Pipeline (CR: FM)	60.34	55.98	58.08
Pipeline (CR: LM)	60.97	56.39	58.59
Joint Learning	61.41	57.40	59.33
Chen et al. (2021)	61.67	57.45	59.49

Table 4: AZP resolution results of *pipeline*, *joint learning* and Chen et al. (2021). FM refers to using the first mention as the cluster representation while LM refers to the last mention.

8 Discussion

The main difference between our *joint learning* approach and Chen et al. (2021) is how AZPs are detected and learned. In our approach, we detect AZPs initially before we cluster them with other mentions, while Chen et al.’s (2021) model learns clustering AZPs and mentions in an end-to-end system. Our results appear to confirm earlier results that considering AZP identification end-to-end in the coreference resolution task can negatively affect the performance on the task (Iida and Poesio, 2011; Chen et al., 2021). One possible explanation might be the overall performance of the mention detection on non-AZPs is better than AZPs (Chen et al., 2021). Chen et al. (2021) consider every gap as a candidate AZP, which increases the space of possible candidates and affects their detection recall. To mitigate this problem, we use a different neural component for AZP detection. The AZP identification that we used in the *joint learning* and *pipeline* settings only

considers gaps that appear after verbs which limits the number of candidates. Moreover, the AZPs in the *joint learning* have explicit tags which might have resulted in their correct detection, which could be why the approach achieved better results. The main limitation of our proposed approaches is if the AZP identification fails to detect many AZPs in the test phase, it might have dropped the evaluation of the coreference resolution and AZP resolution. Pre-training BERT with AZPs can be beneficial. Existing language models (LMs) learn by masking words or perturbing their order (Qiu et al., 2020), but this is not applicable to AZPs. (Konno et al., 2021) have shown two approaches to improve LMs so they work for AZPs, first by introducing a new pre-training task and second by a new fine-tuning technique. They showed an increased performance for AZP resolution for Japanese. In future works, we intend to pre-train a large-scale LM using their methods and see if it can improve the performance of the AZP and coreference resolution tasks.

9 Conclusion

We proposed two architectures to resolve AZPs and non-AZPs jointly. The first approach is in a *pipeline* setting and the second in a *joint learning* representation. The *joint learning* outperformed the *pipeline* and another approach (Chen et al., 2021) in the joint coreference resolution. We also extended the Arabic portion of CoNLL-2012 to include AZPs which will be suitable for future works and shared-tasks that resolves AZPs and non-AZPs together.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful feedback which helped to improve an early version of the paper.

References

- Abdulrahman Aloraini and Massimo Poesio. 2020a. Anaphoric zero pronoun identification: A multilingual approach. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 22–32.
- Abdulrahman Aloraini and Massimo Poesio. 2020b. Cross-lingual zero pronoun resolution. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 90–98.
- Abdulrahman Aloraini, Juntao Yu, and Massimo Poesio. 2020. Neural coreference resolution for arabic. *arXiv preprint arXiv:2011.00286*.
- Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.
- Anders Björkelund and Pierre Nugues. 2011. Exploring lexicalized features for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 45–50.
- Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57.
- 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.
- Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788.
- Shisong Chen, Binbin Gu, Jianfeng Qu, Zhixu Li, An Liu, Lei Zhao, and Zhigang Chen. 2021. Tackling zero pronoun resolution and non-zero coreference resolution jointly. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 518–527.
- Barbara Di Eugenio. 1990. Centering theory and the italian pronominal system. In *Proc. of the 13th COLING*, Helsinki, Finland.
- Mushira Eid. 1983. On the communicative function of subject pronouns in arabic. In *Journal of Linguistics* 19.2, pages 287–303.
- Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.
- 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 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.
- Ganesh Jawahar, Benoît Sagot, and Djamel Seddah. 2019. What does bert learn about the structure of language?

- In *57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy*.
- Megumi Kameyama. 1985. *Zero Anaphora: The case of Japanese*. Ph.D. thesis, Stanford University, Stanford, CA.
- Fang Kong and Hwee Tou Ng. 2013. Exploiting zero pronouns to improve chinese coreference resolution. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 278–288.
- Ryuto Konno, Shun Kiyono, Yuichiroh Matsubayashi, Hiroki Ouchi, and Kentaro Inui. 2021. Pseudo zero pronoun resolution improves zero anaphora resolution. *arXiv preprint arXiv:2104.07425*.
- Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.
- Baoli li. 2012. Learning to model multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL2012-Shared Task*.
- Charles N. Li and Sandra A. Thompson. 1979. Third person pronouns and zero anaphora in chinese discourse. In *Syntax and Semantics*, volume 12: Discourse and Syntax, pages 311–335. Academic Press.
- Ting Liu, Yiming Cui, Qingyu Yin, Weinan Zhang, Shijin Wang, and Guoping Hu. 2017. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. In *arXiv preprint arXiv:1606.01603*.
- Bonan Min. 2021. Exploring pre-trained transformers and bilingual transfer learning for arabic coreference resolution. In *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 94–99.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task. Association for Computational Linguistics, Association for Computational Linguistics.*, pages 1–40.
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.
- Tomohide Shibata and Sadao Kurohashi. 2018. Entity-centric joint modeling of japanese coreference resolution and predicate argument structure analysis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 579–589.
- Marcus Stamborg, Dennis Medved, Peter Exner, and Pierre Nugues. 2012. Using syntactic dependencies to solve coreferences. In *Joint Conference on EMNLP and CoNLL2012-Shared Task*.
- Olga Uryupina, Alessandro Moschitti, and Massimo Poesio. 2012. Bart goes multilingual: the unitt/essex submission to the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 122–128. Association for Computational Linguistics.
- Lydia White. 1985. The “pro-drop” parameter in adult second language acquisition. *Language learning*, 35(1):47–61.
- Jingxuan Yang, Si Li, Sheng Gao, and Jun Guo. 2022. Corefdpr: A joint model for coreference resolution and dropped pronoun recovery in chinese conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:571–581.
- Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2016. A deep neural network for chinese zero pronoun resolution. In *arXiv preprint arXiv:1604.05800*.
- Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017. Chinese zero pronoun resolution with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1318.
- Qingyu Yin, Yu Zhang, Weinan Zhang, Ting Liu, and William Yang Wang. 2018. Zero pronoun resolution with attention-based neural network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 13–23.
- Kim Young-Joo. 2000. Subject/object drop in the acquisition of korean: A cross-linguistic comparison. In *Journal of East Asian Linguistics 9.4*, pages 325–351.
- 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 in Natural Language Processing and Computational Natural Language Learning*, pages 541–550.
- Desislava Zhekova and Sandra Kübler. 2010. Ubiu: A language-independent system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, page 96–99.

A CoNLL-2012 Annotation Layers

The CoNLL-2012 annotation layers consists of the following (Pradhan et al., 2012):

1. Document ID: Contains the file name.
2. Part number: Some files are divided into several files and this number shows the sentence number.
3. Word number: Word position in the sentence.
4. Word itself: This represents the tokenized token.
5. Part-of-Speech: The Part-of-speech of the word.
6. Parse bit: This is the bracketed structure broken before the first open parenthesis in the parse, and the word/part-of-speech leaf is replaced with a *.
7. Lemma: Used to show the gold and predicate lemma.
8. Predicate Frameset ID: This is the PropBank frameset ID of the predicate in Column 7.
9. Word sense: The word sense.
10. Speaker/Author: The speaker or author name, where available. Mostly in broadcast conversation and web log data. However, this is not available for Arabic because all texts are extracted from newspapers.
11. Named Entities: Named entity for the word.
12. Arguments: Predicted and gold arguments.
13. Coreference: Coreference chain which can be single or multiple tokens.