Optimizing Entity Resolution in Voice Interfaces: An ASR-Aware Entity Reference Expansion Approach

Jiangning Chen
Splunk AI
jiangningc@splunk.com

Ziyun Zhang
California Institute of Technology
zyzhang@caltech.edu

Qianli Hu
Xsolla
ql.hu@xsolla.com

Abstract

This paper tackles the challenges presented by Automatic Speech Recognition (ASR) errors in voice-based dialog systems, specifically, their adverse impact on Entity Resolution (ER) as a downstream task. Navigating the equilibrium between accuracy and online retrieval's speed requirement proves challenging, particularly when limited data links the failed mentions to resolved entities. In this paper, we propose a entity reference expansion system, injecting pairs of failed mentions and resolved entity names into the knowledge graph, enhancing its awareness of unresolved mentions. To address data scarcity, we introduce a synthetic data generation approach aligned with noise patterns. This, combined with an ASR-Error-Aware Loss function, facilitates the training of a RoBERTa model, which filters failed mentions and extracts entity pairs for knowledge graph expansion. These designs confront obstacles related to ASR noise, data limitations, and online entity retrieval.

1 Introduction

In the domain of voice-based dialog systems, the inherent inaccuracies within Automatic Speech Recognition (ASR) pose significant impediments to downstream tasks. Specifically, as the transcribed input undergoes processing by a Natural Language Understanding (NLU) component to extract structured data such as entity mentions, errors in ASR frequently propagate to the subsequent component in a dialog system - Entity Resolution (ER). ER is the process of linking labeled mentions to a knowledge base, and the reliance on ASR accuracy exacerbates the intricacy of this task.

Further complicating matters is the imperative for stringent resource optimization, mandated by the latency requirements associated with deploying ER systems on devices. Within this context, the most pragmatic workaround, namely token-based matching, may encounter limitations when confronted with noisy or ambiguous entity mentions. For example, a token-based system might proficiently recognize "Flying Gorilla" but could falter when dealing with semantically or phonetically akin phrases such as "Frying Gorilla" or "Flying Gloria." These challenges often emanate from ASR errors, underscoring the necessity for nuanced solutions in the development of formal voice-based dialog systems.

To address these challenges, this paper introduces an enhanced entity reference enrichment system for the knowledge graph. Our offline model, depicted in Figure 1, utilizes a synthetic data generation pipeline that augments all entity names to replicate error patterns observed in live traffic. This approach addresses data scarcity issues and enables fine-tuning of a RoBERTa-based encoder using a cross entropy loss that is sensitive to ASR-induced inaccuracies, capturing both semantic and phonetic subtleties. The model specifically encodes and generates pairs between previously failed mentions and successfully resolved entity names, which are injected back into the knowledge graph to improve its handling of historically unresolved mentions while maintaining low latency in our industrial retrieval pipeline. Although this approach focuses on errors previously encountered, it captures a significant proportion of common error patterns. While a more dynamic system capable of addressing new ASR errors could involve runtime vector searches, the required infrastructure changes and potential latency impacts make our current method a practical short-term solution, setting the stage for future enhancements.

The contributions of this paper are three-fold: firstly, it introduces a synthetic data generation approach to train the model against upstream patterns of noises and reduce manual labeling. Secondly, our ASR-Error-Aware Loss function enhances the RoBERTa model's performance in handling ASR-

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1–7

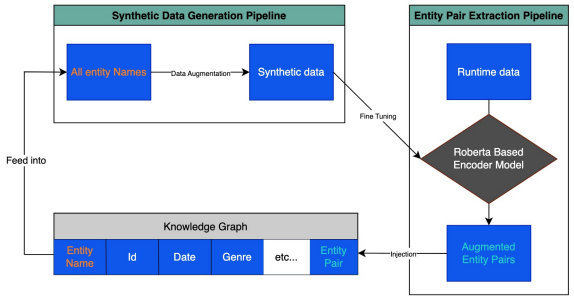

Figure 1: Overview of the offline entity reference expansion system, depicting its two core components: the synthetic data generation and the entity pair extraction pipelines. The data generation pipeline receives the entity names from the knowledge graph and augments them to produce synthetic data, fine-tuning a RoBERTa-based model. The model then encodes and filters the runtime data in the entity pairing pipeline.

induced inaccuracies. Finally, the paper proposes a knowledge graph (KG) injection strategy that can be integrated into the runtime pipeline without modifying the existing online retrieval strategy. These contributions collectively address challenges related to training data, ASR noise, and online processing.

2 Related Work

Early initiatives for entity resolution tasks, such as Neural Entity Linking (NEL) and Entity Disambiguation (ED), utilized fully-connected neural networks or Long Short-Term Memory (LSTM) networks to encode mentions and entity names (Kolitsas et al., 2018; Gillick et al., 2019). The emergence of deep pre-trained models like BERT (Devlin et al., 2018) and their fine-tuned derivatives, marked a paradigm shift in methodologies for Entity Linking and Entity Resolution (Wu et al., 2019; Li et al., 2021). These models typically embed entity mentions and names into a dense vector space, employing architectures such as two-tower designs (Gillick et al., 2019), and calculating the semantic similarity between mentions and entities in the Knowledge base (Ganea and Hofmann, 2017; Raiman and Raiman, 2018).

Vector Search or Nearest Neighbor Search techniques are commonly used for retrieving the best candidate entities. However, they cannot scale in high-latency settings (Li et al., 2021; Zhou et al., 2022). Innovations like the siamese structure with improved alignment networks proposed by Li et al. (Li et al., 2021) aim to reduce exhaustive computations. In contrast, we introduce an offline process for entity pair extraction to minimize online latency demands.

While Wang et al. (Wang et al., 2020, 2021) focused on improving entity retrieval by correcting ASR-induced query errors, we utilize a fine-tuned encoder model, notably RoBERTa, to enhance entity retrieval accuracy by expanding the candidate pool, addressing a different facet of the ER challenge.

The dilemma of scarce labeled data in industrial NLP applications is well-acknowledged, with the lack of manual annotation posing significant constraints. While model transfer and data augmentation are common remedies, our approach leans towards data augmentation. This strategy aligns with our objectives, providing cost-effective control over training data distribution and enabling us to fine-tune our model in a manner that is more reflective of real-world voice-based interactions.

3 Methodology

3.1 Problem Overview

The overview of our system is as follows. Given an entity mention Q by a user, we resolve the corresponding entity name among the entity candidates $\{C_i\}_{i=1}^{m}$ from a knowledge graph; the number of candidates could vary depending on the application setting.

We train a deep encoder model to embed Q and $\{C_i\}_{i=1}^{m}$ in a vector space, and use their similarity scores to rank and select the candidates. To meet the latency constraint, our embedding and scoring are conducted offline. Using the similarity scores, we extract entity pairs with a two-stage filtering process (detailed in Section 3.4.1). The extracted entity pairs are then injected into the knowledge graph for entity reference expansion.

3.2 Encoder Model

In our approach, we employ the RoBERTa model (Liu et al., 2019) to encode mentions and entities. Due to its ability to encapsulate a holistic sentence context, we specifically use the embedding from the CLS token, a special symbol at the start of each input, to represent each entity mention and name in the $\mathbb{R}^{768}$ vector space. This decision is based on our empirical findings where the CLS vector exhibited better performance in entity resolution tasks compared to the average of word embeddings.

The RoBERTa model, powerful in capturing

semantic meanings in generic English text, was pre-trained on massive corpora. Our experiments confirmed that pre-trained RoBERTa without fine-tuning does improve ER quality. However, the pre-trained model is unable to recognize nuanced patterns in some specific domain ER, especially ASR (Automated Speech Recognition) noise. Thus, we need to fine-tune the model over specific domain ER datasets.

As real traffic analysis shows that ASR Score is a strong indicator for potential improvement, we add a penalty term L_{ASR} to the loss function to penalize the loss when the entity mention has ASR errors: $L_{ASR} = e^{(1-\text{ASR\_Score}(C_1))}$. The ASR Score is predicted by the upstream ASR model, ranging between 0 and 1.

Let R be an entity mention. Let $C_1, C_2, \ldots, C_n$ be its entity candidates, where C_1 is the true target (positive candidate) and $C_2, \ldots, C_n$ are negative candidates. Let E_R, E_{C_i} denote the embedded vectors for R and C_i, respectively. Let $\langle E_R, E_{C_i} \rangle$ be the dot-product similarity of the embeddings of query R and candidate C_i. Then the standard Cross Entropy Loss can be defined as:

$$L_{CE} = -\log \frac{e^{\langle E_R, E_{C_1} \rangle}}{e^{\langle E_R, E_{C_1} \rangle} + \ldots + e^{\langle E_R, E_{C_n} \rangle}}$$

Now we introduce the ASR-Error-Aware Loss combined with Cross Entropy Loss, defined as:

$$L_{AEA} := L_{ASR} \cdot L_{CE}$$

One obstacle we face during the encoder model's training is the scarcity of labeled training data. To tackle this issue, we employ data generation techniques to create synthetic entity mentions that resemble the patterns in real user queries, the details of which we will discuss in the next section.

3.3 Data Augmentation with ASR Score Simulation

The training data for fine-tuning the RoBERTa model is generated by data augmentation. Synthetic entity mentions are generated solely from the entity names in the knowledge graph. In this way, the pipeline is not constrained by the lack of human annotations and is protected from data imbalance issues.

The strategy for data augmentation is inspired by the following observation of the live traffic. When comparing a user entity mention with its true entity name, the noises and errors in entity mentions often follow common patterns. Therefore, for each entity name in the knowledge graph, we generate synthetic mentions of the following six types (Figure 2). To obtain ASR scores of synthetic data, we first compute the mean and variance of the ASR scores for each of these types, and then calculate the scores using a normal distribution based on computed mean and variance for each of these types:

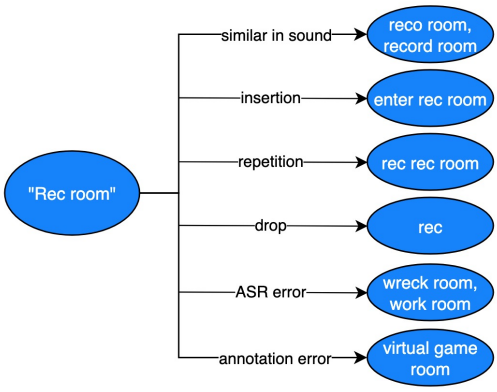

Figure 2: The six types of synthetic mentions derived from entity names based on the error patterns. In this example, we show how to generate synthetic data by the entity name "Rec room."

(1) User replaced a word with another word similar in sound (e.g., “rec room” to “record room”) - this type of data amounts to 14.52% of the total amount of 25k generated data;

(2) Upstream NER error. Inserted common words from the query vocabulary (e.g., “rec room” to “enter rec room”) - 15.19%;

(3) User repeated words (e.g., “rec room” to “rec rec room”) - 26.35%;

(4) Words dropped randomly, which may come from noise (e.g., “rec room” to “rec”) - 25.90%;

(5) ASR error: replaced words with common ASR-confusing words (e.g., “rec room” to “wreck room”) - 11.87%. This helps to train the model to learn phonetic similarities.

(6) Synonym replacement error, the most frequent errors made by annotators (e.g., “rec room” to “virtual game room”) - 6.16%.

Each synthetic mention and its original entity name are then used as the mention R and the true target C_1. The synthetic dataset we are using has been divided into two separate sets: the training set and the validation set. While the synthetic dataset

provides a good foundation for testing our model, we also make use of a smaller, manually annotated test set comprised of historical real traffic data. This manually annotated dataset is particularly useful because it better represents real-world use cases and allows us to ensure the performance of our synthetic dataset more accurately. By combining both synthetic and real data, we prepare our model to be deployed in real-world scenarios.

3.4 Entity Pair Extraction

In this section, we describe how we extract a list of entity pairs in the form of (entity mention, resolved entity name). But first we need to remark that a different way to use the fine-tuned encoder would be to encode each entity mention during runtime, and perform a vector search to select the best candidate. This is however impractical for two reasons: first, vector search could cause significant latency when the number of candidates is large; second, the forward inference of a deep encoder model can be slow and increase latency. In contrast, our framework offloads most of the heavy computation to the offline stage and provides a solution to minimize latency.

3.4.1 Entity Pair Extraction with Model-Based Pairing

The process of entity pair extraction is as follows. We gather the previously failed entity mentions, i.e., the entity mentions that the pre-existing ER system could not resolve. We use the model to compute the embeddings of these failed mentions, and compare with the embeddings of the known entities and successfully resolved mentions. We use a filtering method (discussed in Section 3.4.2) to pair them, and retain only those pairs with high confidence. See Algorithm 1 for detailed pseudocode for the extraction process.

Algorithm 1: Entity pair extraction

```
Data:
S ← task entity pairs from real traffic data
S_1 ← failed task entity pairs in S
S_2 ← successful task entity pairs in S
FM ← failed entity mentions set in S_1
N ← resolved entity names in S_2
SM ← resolved entity mentions in S_2
Load model and embed:
Load the embedding model
Use the model to embed the sets FM, N, SM
Pairing:
for mention ∈ FM do
    Selectively pair with entities in N by
      Algorithm 2 to obtain a pairing
      dictionary D
    Remove duplicates in D
    Generate entity pairs from D
end
Additional Filtering (Optional):
for mention ∈ entity pairs do
    Compute its ratio of historical
      failed/successful cases
    if ratio value < threshold then
        Remove this mention
    end
end
```

3.4.2 Filtering method

We now expand on the crucial filtering stage during the pairing process. We experimented with several different filtering methods and selected a strategy that prioritizes a low regression rate, ensuring minimal disruption to existing data integrity. As described in Algorithm 2, our approach utilizes a two-stage filtering method. Initially, we filter by absolute thresholds on cosine similarity to capture phonetic similarities indicative of ASR errors. Subsequently, we apply a lexical string similarity filter. This second stage is designed to temper the inclusion rate of new reference pairs into the knowledge graph, preventing an overly aggressive expansion that could impact runtime performance adversely. As depicted in Figure 3, this dual-stage approach ensures a balanced enhancement of the knowledge graph's accuracy. If ongoing evaluations indicate stable performance improvements, we plan to phase out the lexical similarity filtering, shifting towards a more dynamic and phonetically-focused expansion strategy in future iterations.

4 Experiments and Results

4.1 Training Setup

We implemented the RoBERTa model in PyTorch (Paszke et al., 2019), initializing it with the pre-trained RoBERTa base (Liu et al., 2019). Similarly, we also implemented the SentenceTransformer all-mpnet-base-v2 model (Reimers and Gurevych, 2019), starting with its pre-trained version. Both models were optimized using the Adam optimizer (Kingma and Ba, 2014) with weight decay (Loshchilov and Hutter, 2018). The learning

Algorithm 2: Filtering

for *each resolved mention* $SM_i \in SM$ **do**
  Filter the failed mentions in FM by their cosine similarity with the entity mention SM_i: cos_sim$(FM_j, SM_i) >$ emb_sim_threshold;
  for *each of the remaining failed mentions* **do**
    Filter by their lexical string similarity with the resolved entity name N_i: lexical_sim$(FM_j, N_i) >$ str_sim_threshold;
  end
end

rate was set to 10^{-5}, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a batch size of 64. Training and testing split is 80:20.

4.2 Evaluation of Encoder Models

We assessed the performance of various pre-trained models, including Google's text-bison and OpenAI's text-embedding-ada, alongside fine-tuned SentenceTransformer (ST) and RoBERTa models. The evaluation dataset consisted of 328 widely-used entity names and approximately 1000 related entity mentions. Ranking of entity mention candidates was based on cosine similarity within the embedding space.

Recall metrics at 1 (r1), 3 (r3), and 6 (r6) positions, along with the Mean Reciprocal Rank (MRR), were computed for both the pre-trained and fine-tuned versions under two different loss functions: the standard Cross Entropy loss (L_{CE}) and our proposed ASR-Error-Aware Loss (L_{AEA}). These metrics were calculated relative to a baseline that utilized lexical similarity-based search.

Model	r6(%)	r3(%)	r1(%)	MRR(%)
Pre-trained ST	6.79	6.21	6.02	6.15
Pre-trained RoBERTa	6.73	6.21	5.99	6.14
text-bison	6.83	6.59	6.16	6.38
text-embedding-ada	8.24	7.69	7.33	7.50
ST+L_{CE}	52.25	46.99	46.12	47.03
RoBERTa+L_{CE}	52.52	47.13	46.92	47.08
ST+L_{AEA}	52.73	47.75	46.61	47.42
RoBERTa+L_{AEA}	53.23	47.89	46.77	47.69

Table 1: Relative improvement of encoder models under different configurations and loss functions.

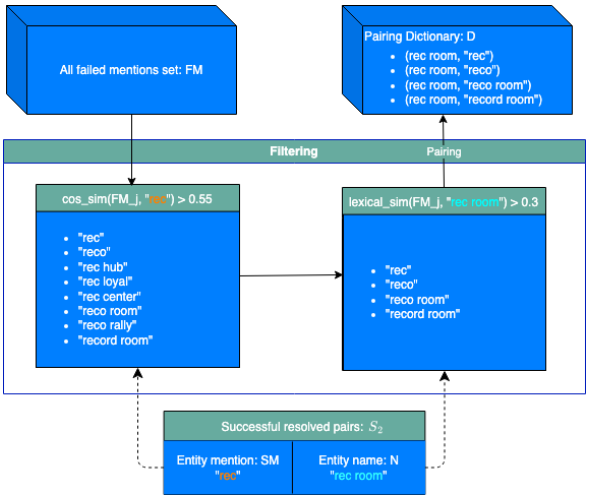

Figure 3: A demonstration of how the failed mentions in FM are filtered by one resolved pair (entity mention, entity name) in S_2, in a two-stage process. After filtering, we pair the filtered mentions with the resolved entity name ("rec room" in this case) and put them into our preliminary pairing dictionary D.

The experimental findings conclusively show that the fine-tuned RoBERTa model with the ASR-Error-Aware Loss function (L_{AEA}) yields the best performance. Among pre-trained models, Google's text-bison and OpenAI's text-embedding-ada exhibit superior performance over their counterparts. However, due to their significantly larger architectures, fine-tuning these embeddings is not feasible for on-device applications, where model size and efficiency are crucial.

RoBERTa's dominance over SentenceTransformer in our experiments can be attributed to several factors. Firstly, RoBERTa's pre-training process, involving dynamic masking and training on a larger, more diverse dataset, provides a more nuanced understanding of language. This depth is particularly beneficial in handling the complexities of ASR errors. Furthermore, RoBERTa benefits from extended training periods and additional optimization steps, allowing it to develop a more sophisticated language model. Another critical aspect is the nature of the input data. SentenceTransformer, originally trained for comparing the similarity of longer text segments, may not be as adept at processing the shorter phrases typically seen in our use case. In contrast, RoBERTa's training and architecture make it more suitable for accurately capturing and processing the semantic and phonetic variations present in these shorter utterances. These factors collectively contribute to RoBERTa's en-

hanced performance in our entity resolution tasks.

4.3 Offline Testing

Next, we test the ER system, which comprises both the trained encoder and the entity pair extraction pipeline, using historical real traffic data. Entity pairs are extracted according to Algorithm 1 with the fine-tuned encoder. To evaluate system improvements, we compute a win/loss ratio, where a "win" represents previously failed queries now resolved by the system, and a "loss" represents previously successful queries that are erroneously resolved by the new system. A higher win/loss ratio indicates better system performance, combining the previously separate "fixed ratio" and "regression ratio" into a single, more interpretable metric.

Table 2 shows the results of offline testing without the optional filtering step in Algorithm 1, while Table 3 presents results with the filtering step applied. The filtering thresholds are set at 0.55 for cosine similarity and 0.3 for lexical similarity. It is observed that the fine-tuned encoder model achieves significant improvements over both the baseline and the pre-trained model without fine-tuning. The optional filtering step, while reducing the regression ratio, does so at the cost of a lower fixed ratio, now combined into the win/loss ratio. The decision to include the filtering step depends on the specific needs and constraints of the application setting.

Method	**Metric**	**Result (%)**
RoBERTa Pre-trained	Fixed ratio	33.12
	Regression ratio	1.38
	Win/Loss ratio	24
RoBERTa+L_{CE}	Fixed ratio	37.48
	Regression ratio	0.65
	Win/Loss ratio	57.66
RoBERTa+L_{AEA}	Fixed ratio	37.92
	Regression ratio	0.65
	Win/Loss ratio	58.34

Table 2: Offline testing of fine-tuned model versus pre-trained (no filtering)

4.4 Online Testing

Finally, we evaluate the ER system with two large-scale AB experiments on live traffic. The experiment results are mainly for two top domains in voice assistant scenarios. We measure the results in three metrics: task success rate, failed task count, and end-to-end latency.

The first AB test was conducted with the following setting: the test group uses the *pre-trained RoBERTa* entity reference expansion solution in ER; the control group shows default prod behavior without entity reference expansion. The second AB test was conducted with the following setting: the test group uses the *fine-tuned RoBERTa+*L_{AEA} entity reference expansion solution in ER; the control group uses the pre-trained RoBERTa entity reference expansion solution in ER. Both AB experiments have been running for 2 weeks for observation.

Method	**Metric**	**Result (%)**
RoBERTa Pre-trained	Fixed ratio	19.06
	Regression ratio	1.11
	Win/Loss ratio	17.17
RoBERTa+L_{CE}	Fixed ratio	26.59
	Regression ratio	0.52
	Win/Loss ratio	51.13
RoBERTa+L_{AEA}	Fixed ratio	26.86
	Regression ratio	0.48
	Win/Loss ratio	55.96

Table 3: Offline testing of fine-tuned model versus pre-trained (with filtering)

Table 4 and 5 show the relative improvement of the pre-trained model versus no entity reference expansion ER and fine-tuned model versus pre-trained model. The improvement can be seen from two aspects: (1) fewer instances of failed tasks, which means we were able to resolve entities more frequently instead of sending the failed resolved strings as a store search; (2) an increase in user confirmation that task is successfully resolved. The results indicate that the new treatment has a significant positive impact on the task success rate without much sacrifice in end-to-end latency.

Task	**Metric**	**Result**
All Device	Task success rate	$+1.23\%$
Target tasks	Task success rate	$+2.41\%$
All Device	Failed task count	-10.06%
Target tasks	Failed task count	-13.51%
E2E	Latency	$+0.4ms$

Table 4: First online AB testing: pre-trained model versus no entity reference expansion ER

Table 6 gives some illustrative examples comparing the ER results with and without using the proposed entity reference expansion. It can be observed that our approach can effectively resolve noisy entity mentions by capturing semantic or phonetic similarities that the default matching-based ER system cannot handle.

Task	Metric	Result
All Device	Task success rate	$+1.19\%$
Target tasks	Task success rate	$+2.33\%$
All Device	Failed task count	-10.3%
Target tasks	Failed task count	-13.9%
E2E	Latency	$+0.07ms$

Table 5: Second online AB testing: fine-tuned model versus pre-trained model

Entity Mentions	Groundtruth	Former ER	New ER
"super fly game"	superfly	[]	[superfly]
"fly girl"	flying gorilla	[]	[flying gorilla]
"best star"	beatstar	[]	[beatstar]
"president evil four"	resident evil 4	[]	[resident evil 4]

Table 6: Examples of entity mentions that the new ER system (with the proposed entity reference expansion) can resolve while the former token-matching based ER fail to resolve.

5 Conclusion

In conclusion, our entity reference expansion pipeline, utilizing a fine-tuned RoBERTa model, seeks to enhance Entity Resolution (ER) in voice-based conversational systems. The synthetic data generation approach, which emulates noise patterns, facilitates model training without requiring manual labeling, while the implementation of an ASR-Error-Aware Loss function addresses challenges arising from ASR-induced noise. Furthermore, our knowledge-graph-injection approach, executed offline, ensures the system's robustness while seamlessly aligning with the industry's online retrieval design for swift performance. Our developments offer new perspectives in enhancing ER solutions, contributing to the ongoing improvement of voice-based dialog systems.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. *CoRR*, abs/1704.04920.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. *arXiv preprint arXiv:1808.07699*.

Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for bert-based entity resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, pages 13226–13233.

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

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Jonathan Raiman and Olivier Raiman. 2018. Deeptype: multilingual entity linking by neural type system evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Haoyu Wang, John Chen, Majid Laali, Jeff King, Kevin Durda, William M. Campbell, and Yang Liu. 2021. Leveraging asr n-best in deep entity retrieval. In *Interspeech 2021*.

Haoyu Wang, Shuyan Dong, Yue Liu, James Logan, Ashish Agrawal, and Yang Liu. 2020. Asr error correction with augmented transformer for entity retrieval. In *Interspeech 2020*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2019. Scalable zero-shot entity linking with dense entity retrieval. *arXiv preprint arXiv:1911.03814*.

Xiaozhou Zhou, Ruying Bao, and William M. Campbell. 2022. Phonetic embedding for asr robustness in entity resolution. In *Interspeech 2022*.