# EXPLORING NEURAL ENTITY LINKING WITH PRE-TRAINED TRANSFORMER MODELS FOR TURKISH

by
FATIH BEYHAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of Master of Sciences

Sabancı University
July 2023

## ABSTRACT

EXPLORING NEURAL ENTITY LINKING WITH PRE-TRAINED TRANSFORMER MODELS FOR TURKISH

FATIH BEYHAN

COMPUTER SCIENCE AND ENGINEERING M.SC. THESIS, JULY 2023

Thesis Supervisor: Prof. Ayşe Berrin YANIKOĞLU

Keywords: Entity Linking, Transformer Models, Transfer Learning

Entity Linking, a vital component of Natural Language Processing (NLP), aims to link named entities in a given text to their corresponding real-world entities in a knowledge base. This work presents an exploration of transformer-based Neural Entity Linking models adapted to the Turkish language, focusing on their robustness across genre and domain shifts.

We take two advanced Neural Entity Linking models originally designed for English and adapt them to Turkish. We then thoroughly assess how well these adapted models perform on different Turkish datasets, along with a new method we developed called EntityBERT, which serves as a reference point for comparison. EntityBERT is a simple Neural Entity Linking model which exploits pretrained Turkish transformer model and contextualized learning capabilities of transformer models.

The evaluation was conducted on three distinct datasets, including one newly created dataset, publicly available for further research. The findings revealed that Neural Entity Linking models exhibited robust performance across language and genre shifts, demonstrating their adaptability to Turkish and diverse textual genres. Nonetheless, our investigation also highlights a noteworthy limitation: the susceptibility of Neural Entity Linking models to domain shift challenges. Despite their favorable performance in general settings, adapting to domains with distinctive characteristics poses considerable difficulties.

Overall, this study sheds light on the potential and limitations of Neural Entity Linking models in Turkish, provides an evaluation dataset of Turkish tweets, and finally delivers valuable insights for advancing the field of natural language processing in non-English languages.

# ÖZET

TÜRKÇE İÇİN ÖNCEDEN EĞİTİLMİŞ DÖNÜŞTÜRÜCÜ MODELLERİYLE SİNİRSEL VARLIK İLİŞKİLENDİRME PROBLEMİNİN KEŞFEDİLMESİ

FATİH BEYHAN

BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, TEMMUZ 2023

Tez Danışmanı: Prof. Ayşe Berrin YANIKOĞLU

Anahtar Kelimeler: Varlık İlişkilendirme, Dönüştürücüler Modeller, Öğrenme Aktarımı

Doğal Dil İşleme'nin hayati bir bileşeni olan Varlık İlişkilendirme, belirli bir metindeki adlandırılmış varlıkları bir bilgi tabanındaki karşılığı olan gerçek dünya varlıklarına bağlamayı amaçlar.

Bu çalışma, Türk diline uyarlanmış Dönüştürücü tabanlı Sinirsel Varlık İlişkilendirme modellerinin verisetlerindeki yazım tipi ve tür değişimlerindeki performansına odaklanarak bir araştırma sunmaktadır. Orijinal olarak İngilizce için tasarlanmış iki gelişmiş Sinirsel Varlık İlişkilendirme modeli Türkçe'ye uyarlanmıştır. Daha sonra, uyarlanan bu modellerin farklı Türkçe veri setlerinde performansları incelenmiştir. Ayriyetten, EntityBERT adlı geliştirdiğimiz ve karşılaştırma için bir referans noktası işlevi gören yeni bir yöntem ayrıntılı olarak değerlendirilmiştir. EntityBERT, önceden eğitilmiş Türkçe Dönüştürücü modelini ve bu modellerin 'bağlamsal öğrenme' (contextual-learning) yeteneklerini kullanan basit bir Sinirsel Varlık İlişkilendirme modelidir.

Sistemlerin performans değerlendirmesi tarafımızca yeni oluşturulmuş bir veri seti de dahil olmak üzere üç farklı veri seti üzerinde gerçekleştirilmiştir. Bulgular, Sinirsel Varlık İlişkilendirme modellerinin, dil ve içerik türü geçişlerinde güçlü performans sergilediğini ve çeşitli metin türlerine uyum sağlayabildiklerini ortaya koymuştur. Bununla birlikte, araştırmamız kayda değer bir dezavantajı da vurgulamaktadır: Sinirsel Varlık İlişkilendirme modelleri yazım tipi (Vikipedi metinleri ve Tweetler) değişiklikleri karşısında performans kaybetmektedir.

Genel olarak, bu çalışma, Türkçe'deki Sinirsel Varlık İlişkilendirme modellerinin potansiyeline ve dezavantajlarına ışık tutmakta, Türkçe Tweet'lerden oluşan bir değerlendirme seti sunmakta ve son olarak, İngilizce dışındaki dillerde NLP alanında ilerleme sağlamak için değerli bilgiler sunmaktadır.

**ACKNOWLEDGEMENTS**

I would like to thank my current and previous thesis advisors Prof. Ayşe Berrin YANIKOĞLU and PhD. Reyyan YENİTERZİ for pushing me through the course of my Master's studies.

My biggest thanks to m**y** family for their love and **s**upp**o**rt during this long journey of research.

*July 2023, Istanbul*
*Fatih Beyhan*

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

BERT: Bidirectional Encoder Representations from Transformers

CNN: Convolutional Neural Network

EL: Entity Linking

LSTM: Long Short-Term Memory

LUKE: Language Understanding with Knowledge-based Embeddings

ML: Machine Learning

MLM: Masked Language Modelling

NER: Named Entity Recognition

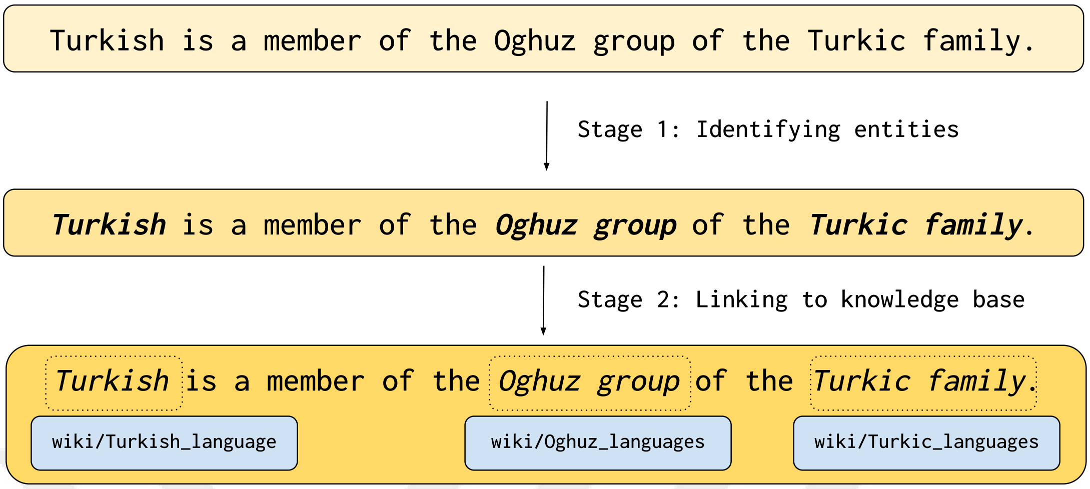
RNN: Recurrent Neural Network

# 1. INTRODUCTION

With the exponentially increasing usage of the internet, the amount of data we can use is growing every day. This vast amount of data is unstructured and comes from various resources such as news websites, blogs, and social media posts like tweets. Even though the data we can reach is growing every day, it is mostly unstructured. To exploit this data, we need systems to shape it into a relatively more structured form. There are many Natural Language Processing and Information Extraction tasks that aim to leverage the usage of unstructured data for text-oriented systems. Entity Linking is one of these tasks.
Entity Linking plays a crucial role in Natural Language Understanding (NLU) and Information Extraction tasks, such as semantic search, entity coreference resolution, question answering, and text summarization. Additional to the classic contribution to Natural Language Processing and Information Extraction, solving the Entity Linking task will lead systems to a more coherent state in terms of understanding the world around them in the era of Large Language Models.
Despite the increasing attention given to Entity Linking in recent years, there has been limited research on Turkish Entity Linking. Even though a very similar task, Word Sense Disambiguation, is widely studied, we cannot say the same thing for Entity Linking in Turkish.
Entity Linking is the process of mapping entities in the given text to their corresponding entry in a knowledge base. In Figure 1.1, you can see an example of Entity Linking process at a very high level. As it can be seen in the figure, Entity Linking can be performed in two stage. In the first stage we identify the named entities and in the second stage we link the identified entities to a knowledge base entry. This is a form of end-to-end Entity Linking, however, it can be performed other-wise as well. In the non-end-to-end version, we try to perform the second stage only where the named entities are already identified.

The terminology for this task has multiple other names like Entity Disambiguation and Entity Resolution where they can be used interchangeably for non-end-to-end Entity Linking. One other name used for this task is called Wikification where the

Figure 1.1 An example going through an end-to-end Entity Linking system. In Stage 1, the raw text input is processed for identifying entities. And in Stage 2, identified named entities are linked to the knowledge base, Wikipedia.




target knowledge base is Wikipedia. However, there are no strict lines between the usage of these names for Entity Linking tasks.

In this work, we are approaching Turkish Entity Linking in a non-end-to-end form. We are exploring two popular Neural Entity Linking models originally designed for English by adapting them for Turkish. Additionally, we are proposing a simplified Neural Entity Linking model, EntityBERT, which exploits the contextualized learning capabilities of transformer models.
We are evaluating these models on different Turkish datasets including one that is created by us. We are providing a small Entity Linking dataset for Turkish tweets that are annotated by us for evaluation purposes. We assessed aforementioned Neural Entity Linking models on three different Turkish datasets which are varying in terms of text genre and domain.

## 1.1 Research Questions

In this thesis, we are exploring different types of state-of-the-art Neural Entity Linking models with different datasets. With these experiments, we are trying to find answers to a few research questions for Neural Entity Linking in Turkish. These questions are as follows;

**RQ1:** Which type of Neural Entity Linking model, either dense-retrieval centric or contextualized-learning centric, would demonstrate superior performance on a small-scale Turkish dataset?

**RQ1.1:** How would these models be influenced by domain and genre shifts when evaluated on different datasets?

**RQ1.2:** Is it feasible to leverage the contextualized learning capabilities offered by transformer models, along with the Masked Language Modeling task, to tackle Neural Entity Linking in the Turkish language?

## 1.2 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 gives a summary of the Entity Linking task's background and describes the previous work done on Neural Entity Linking in general, and Entity Linking for Turkish. Then Chapter 3 provides information regarding the datasets that were used for training and evaluation during this thesis. Chapter 4 explains the technologies, like transformer-based models, that we used. Chapter 5 describes the experiments in detail and provides results. Finally, in Chapter 6, we discuss the results and conclude this thesis by summarizing the research findings and future research directions.

# 2. RELATED WORK

In this chapter, we delve into the antecedent research on the Entity Linking problem, providing the background. We explore investigations that leverage classical Machine Learning techniques along with early Deep Learning models such as RNNs, LSTMs, and CNNs. Finally, we delve into studies that employ Transformer models. This segment is divided into two parts: 1) EL models that integrate Transformer architecture as a component within their pipeline, and 2) EL models that exclusively employ Transformer models.

Previously, researchers used things like name dictionary-based techniques, alias tables, and TF-IDF-based methods (Ratinov, Roth, Downey & Anderson (2011a), Guo, Chang & Kiciman (2013), Bunescu & Paşca (2006), Cucerzan (2007a)). Recent works achieved state-of-the-art results with neural-based models.

One of the most straightforward approaches to Entity Linking is transforming it into a multi-class classification task. In this setting, each mention is a single sample and the Entity Linking model has to classify each of them into one class out of N classes where each of them corresponds to a unique entity. However, systems end up with too many classes to deal with in this setting (Kar, Reddy, Bhattacharya, Dasgupta & Chakrabarti (2018)).

The streamlined approach to the Entity Linking problem is to approach it as a ranking problem. If it is being approached for end-to-end Entity Linking, there are three main tasks; mention detection, candidate generation, and reranking. However, within the scope of this work, we will not focus on end-to-end systems therefore only the last two tasks are relevant.

In the candidate generation task, the EL system returns a set of entities for each entity mention that are treated as a candidate label. And in the reranking task, the EL system receives the generated candidates and reorders them to put the gold label in first order or achieve a better ordering in terms of relevancy.

To generate and rank the candidates, neural networks have been employed by various studies, such as He, Liu, Li, Zhou, Zhang & Wang (2013), Sun, Lin, Tang, Yang, Ji & Wang (2015), Yamada, Shindo, Takeda & Takefuji (2016), Ganea & Hofmann (2017), and Kolitsas, Ganea & Hofmann (2018), which have achieved state-of-the-

art results by modeling context words, spans, and entities.
Logeswaran, Chang, Lee, Toutanova, Devlin & Lee (2019) proposed the zero-shot entity linking task, utilizing cross-encoders for entity ranking but relying on traditional information retrieval techniques for candidate generation. On the other hand, Gillick, Kulkarni, Lansing, Presta, Baldridge, Ie & Garcia-Olano (2019) demonstrated the effectiveness of dense embeddings for candidate generation but did not use pre-training and limited their linking to entities within Wikipedia by incorporating external category labels into their bi-encoder architectures. However, Wu, Petroni, Josifoski, Riedel & Zettlemoyer (2020) (BLINK) combines and extends both of these lines of research, revealing that pre-trained zero-shot architectures are highly accurate and computationally efficient at scale. It utilizes pre-trained transformer models for extracting the dense representations. This model is the state-of-the-art for the dense-retrieval-based models, to our knowledge.
In recent studies, various Transformer-based models have been proposed for Entity Linking. These models leverage the Transformer architecture, originally introduced by Vaswani et al. (2017), and are trained on large entity-annotated corpora from sources like Wikipedia. Broscheit (2019), Ling, FitzGerald, Shan, Soares, Févry, Weiss & Kwiatkowski (2020), Févry, FitzGerald, Soares & Kwiatkowski (2020), Cao, Izacard, Riedel & Petroni (2021), and Barba, Procopio & Navigli (2022) are among the researchers who have explored this approach.
Broscheit (2019) developed an Entity Linking model using BERT as a token classifier by classifying each word in the document with its corresponding entity. Similarly, Févry et al. (2020) used BERT for Entity Linking, but they classified mention spans instead of individual words. Ling et al. (2020) trained BERT to predict entities based on document-level representations. Cao et al. (2021) employed Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Stoyanov & Zettlemoyer (2019) (BART) for Entity Linking, generating referent entity titles for target mentions autoregressively. Barba et al. (2022) treated Entity Linking as a text extraction problem, using BART and Longformer (Beltagy, Peters & Cohan (2020)) to disambiguate mentions by extracting the corresponding referent entity titles. Notably, these models relied solely on local contextual information.
In parallel, other studies like Zhang, Han, Liu, Jiang, Sun & Liu (2019), Yamada, Asai, Shindo, Takeda & Matsumoto (2020); Sun, Shao, Qiu, Guo, Hu, Huang & Zhang (2020) have proposed Transformer-based models that treat entities as input tokens to enrich their expressiveness with additional information contained in the entity embeddings. However, these models were designed for general NLP tasks and have not been specifically tested on Entity Linking. In contrast, Yamada, Washio, Shindo & Matsumoto (2022a) treats entities as input tokens with the specific goal of capturing global context, which has shown to be highly effective for Entity Linking.

In the scope of the Turkish language, there is only a single study related to Entity Linking. Kalender & Korkmaz (2018) worked on this problem by proposing a system that maps the inputs to a Turkish Dictionary or Turkish Wikipedia with a focus on the first one. In other words, while they are approaching the Turkish Entity Linking problem, their focus is more on the Word Sense Disambiguation task. The system they deliver uses linguistic features and simple deep-learning methods for encoding the inputs into dense vectors for a similarity check.

# 3. DATASETS

In this section, we describe the datasets we utilized for training and evaluation with quantitative information. There is one dataset for training and three datasets for evaluation purposes. The training dataset is constructed from the Turkish Wikipedia dump that has the training, development, and test sets. The Mewsli-9 dataset (Botha, Shan & Gillick, 2020), which is publicly available is used only for evaluation. The genre for the aforementioned datasets is different since one of them consists of Wikipedia articles and the other one consists of WikiNews articles. However, both datasets are sharing a formal writing style. Therefore, for an extended evaluation, we created another entity linking datasets which consists of Turkish tweets. These datasets help us to evaluate the robustness of different Entity Linking systems on domain and content shifts. The details of these datasets are explained in the further sub-sections.

## 3.1 Turkish Wikipedia

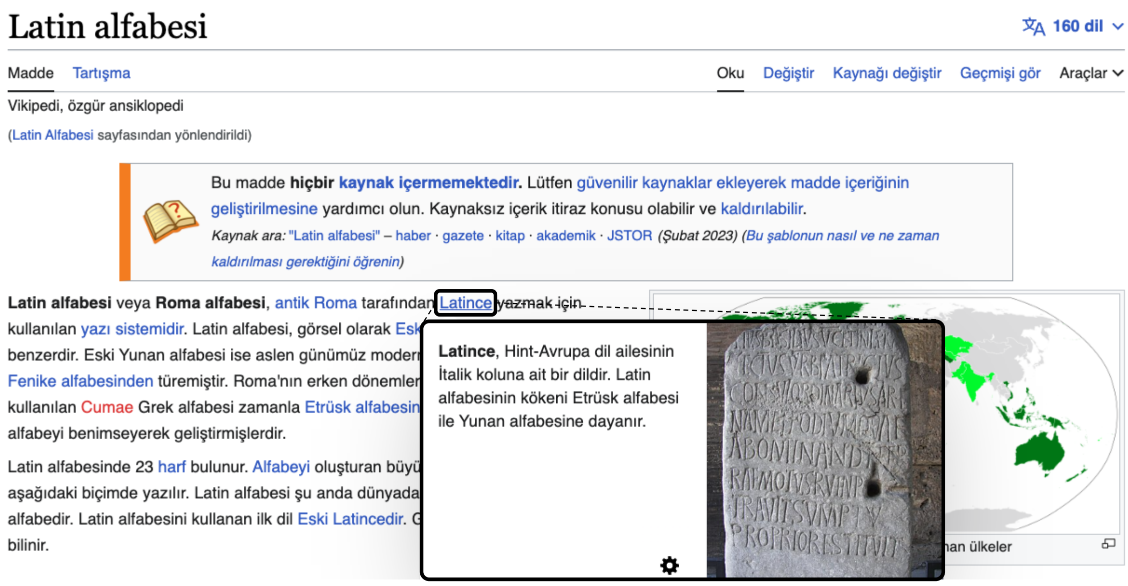
Wikipedia is an open-source multilingual encyclopedia created and maintained by a community of volunteers with millions of daily users. According to Wikipedia, it has more than 61 million articles in 332 different languages. Each page has information about an entity, event, etc. which users write. English is the leading language with 6,683,815 articles, and Turkish is the 29th language with 527,585 articles as of July 15, 2023[1].

Wikipedia is a tailored resource for the entity linking problem due to its rich hyperlink network (interwikies) between different pages. Interwikies are the hyperlink

[1] https://en.wikipedia.org/wiki/List_of_Wikipedias

network of Wikipedia pages. An interwiki can be between two Wikipedia pages. For the source page, where the hyperlink exists, it is called **in-interwiki**, and for the target page, where the hyperlink points to, is called **out-interwiki**. On average, there are 10.55 interwikies per page in Turkish Wikipedia. In Figure 3.1, a part of the Turkish Wikipedia page for Latin Script[2] (*Latin Alfabesi* in Turkish) can be seen. In the first sentence, there is an interwiki for the Turkish Wikipedia page of the Latin[3] (*Latince* in Turkish). This example can be counted as an out-interwiki for the *Latin Script* page and as an in-interwiki for the *Latin* page. As this example shows, these connections are being used as the entity mention for many Entity Linking datasets.

Figure 3.1 A partial screenshot of the Turkish Wikipedia page for Latin Script (*Latin Alfabesi* in Turkish). As the figure shows, there are several interwikies to other Turkish Wikipedia pages including, Ancient Rome, Latin, and more.




Since it is open-source, Wikipedia dumps can be easily downloaded and parsed in order to create an Entity Linking dataset. We have implemented a Wikipedia dump parsing script for the Turkish Wikipedia dump. It uses the `wikipedia2vec`[4] package and saves each page with its content and entity mentions.

When we consider the Wikipedia dump of the 1st of May 2023, we see that there are more than a million pages in total, including pages that are used for redirection, category, disambiguation, and with no content. For instance, there is a page with the URL of `https://tr.wikipedia.org/wiki/MHP`.

[2]https://tr.wikipedia.org/wiki/Latin_alfabesi

[3]https://tr.wikipedia.org/wiki/Latince

[4]https://github.com/wikipedia2vec/wikipedia2vec

However, this is a redirection page and it redirects to the page of *Milliyetçi Hareket Partisi*, a political party abbreviated as *MHP*, with the URL of `https://tr.wikipedia.org/wiki/Milliyet%C3%A7i_Hareket_Partisi`. Also an example of a disambiguation page for the term *Batman* can be found in the Appendix A, Figure A.

After we parsed the dump, there are 564,428 pages with content. 33,752 of these pages had no out-interwiki connection to any other page and 126,161 pages did not have any in-interwiki connection. Table 3.1 shows the top 15 Turkish Wikipedia pages that have the most in-interwiki connections.

Table 3.1 Top most 15 frequent pages that have the most in-interwiki connections and their frequencies.

| **Title** | **Entity ID** | **Wikidata ID** | **Freq.** |
|---|---|---|---|
| Cins | ENT#116457 | Q34740 | 35,220 |
| Amerika Birleşik Devletleri | ENT#46005 | Q30 | 32,246 |
| Mahalle | ENT#645602 | Q123705 | 23,807 |
| Futbol | ENT#218701 | Q2736 | 23,246 |
| İstanbul | ENT#908651 | Q406 | 22,327 |
| Türkiye | ENT#819357 | Q43 | 20,283 |
| Türkiye'nin köyleri | ENT#820553 | Q1529096 | 18,268 |
| Türkler | ENT#820622 | Q84072 | 14,887 |
| Tür | ENT#818801 | Q7432 | 14,874 |
| Almanya | ENT#42662 | Q183 | 14,049 |
| Fransa | ENT#216441 | Q142 | 13,822 |
| Osmanlı İmparatorluğu | ENT#711913 | Q12560 | 13,008 |
| İtalya | ENT#909649 | Q38 | 12,694 |
| Cumhuriyet Halk Partisi | ENT#123205 | Q19079 | 11,620 |
| Ankara | ENT#50403 | Q3640 | 10,023 |

During the parsing, we used `nltk`[5] package for Turkish sentence splitting. The parsing script splits each page into sentences and collects the entity mentions in each sentence into a list. We also create mapping files between the entity id to the title, the entity id to the Wikidata id, etc. Mapping the entity ids to the Wikidata is crucial for being able to use other datasets like Mewsli-9. These datasets are created in different time periods and we used Wikidata as the reference point for mapping these datasets to each other.
After parsing and saving these pages, we work on the sentence level for train, validation, test set splitting, and further processing for creating datasets/dataloaders.

[5] https://www.nltk.org/

Table 3.2 Statistics for the top N frequent pages which has the most interwiki$_{\text{in}}$. **#sents.$_{\textbf{ent.}}$**, **#ent. ment.**, and **avg.** stands for the number of sentences with entity mentions, number of entity mentions and the average number of entity mentions per sentence, respectively.

| **Top N** | **#sents.$_{\textbf{ent.}}$** | **#ent. ment.** | **avg.** |
|---|---|---|---|
| 100 | 586,729 | 703,233 | 1.2 |
| 500 | 959,479 | 1,269,703 | 1.32 |
| 1,000 | 1,175,074 | 1,607,429 | 1.37 |
| 2,000 | 1,412,881 | 2,014,155 | 1.43 |
| 5,000 | 1,774,787 | 2,650,352 | 1.49 |
| 10,000 | 2,066,431 | 3,196,899 | 1.55 |
| 50,000 | 2,789,768 | 4,577,989 | 1.64 |
| 100,000 | 3,085,425 | 5,135,984 | 1.66 |
| 564,428 (ALL) | 3,616,815 | 5,957,791 | 1.65 |

The training, validation, and test sets for training and evaluation purposes are being created on-the-fly with all necessary preprocessing steps.

We have a hyperparameter for the number of entities we would like to use during dataset creation. If it is set to N, then we only use the sentences that have an entity mention for one of the entities in the top N frequent entities. Table 3.1 shows the statistics for top N frequent entities with different N values. As this table shows, when the top 5000 entities are taken almost 1/2 of all entity mention sentences are covered, and this ratio goes down to 1/3 when we take the top 2000 entities.

The dataset splitting into the train, validation, and test sets is happening right before each training starts. We set a ratio for both validation and test sets. We collect all sentences for each entity. We iterate over all entities that we are interested in. We find the number of sentences that a given entity mentioned and then find the number of sentences we want to assign for each set given the ratios we have for validation and test sets. Then, we iterate its sentences and randomly assign the sentence to a set, excluding the sets that we reached the number we desire.

### 3.2 Mewsli-9 Turkish Set

Mewsli-9 (***M**ultilingual **E**ntities in **N**e**ws**, **li**nked*) is a large multilingual Entity Linking dataset that was built and shared by one of the recent works from Google (Botha et al., 2020). It has 289,087 mentions from 58,717 news articles of WikiNews[6]. WikiNews is a free-content news wiki and a Wikimedia Foundation initiative that uses collaborative journalism. The corpus contains news from 9 different languages, including Turkish, across 5 different language families. Table 3.3 shows statistics for the Mewsli-9 dataset across languages.

Table 3.3 Number of documents, entity mentions, and unique entities for the 9 languages of the Mewsli-9 evaluation dataset.

| **Lang.** | **#Docs** | **#Mentions** | **#Unique** |
|---|---|---|---|
| Japanese | 3,410 | 34,463 | 13,663 |
| German | 13,703 | 65,592 | 23,086 |
| Spanish | 10,284 | 56,716 | 22,077 |
| Arabic | 1,468 | 7,367 | 2,232 |
| Serbian | 15,011 | 35,669 | 4,332 |
| **Turkish** | **997** | **5,811** | **2,630** |
| Persian | 165 | 535 | 385 |
| Tamil | 1,000 | 2,692 | 1,041 |
| English | 12,679 | 80,242 | 38,697 |
| Total | 58,717 | 289,087 | 82,162 |

The authors shared the data publicly[7] in the following format; there are text documents for each WikiNews article, and there is a `CSV` file that has the list of all entity mentions with the corresponding Document ID and position with the Wikidata ID. Table 3.2 shows a small portion of the `CSV` data for Turkish. Also, the content of a sample text document of the Mewsli-9 Turkish set can be seen in Appendix A. It is a trivial process to parse the data for further usage after receiving these files.

Table 3.4 The first 4 rows of Mewsli-9 Turkish `CSV` file.

| **Doc ID** | **Pos.** | **Len.** | **Ment. Text** | **URL** | **Wikidata ID** |
|---|---|---|---|---|---|
| tr-6 | 146 | 19 | Türkiye Cumhuriyeti | `wiki/Türkiye` | Q43 |
| tr-6 | 169 | 22 | Ermenistan Cumhuriyeti | `wiki/Ermenistan` | Q399 |
| tr-7 | 89 | 17 | Amerikan Kongresi | `wiki/ABD_Kongresi` | Q11268 |
| tr-7 | 315 | 9 | başörtüsü | `wiki/Başörtüsü` | Q2002752 |

[6]`www.wikinews.org`, using the 2019-01-01 snapshot from archive.org

[7]`https://github.com/google-research/google-research/tree/master/dense_representations_for_entity_retrieval/mel`

Mewsli-9 is using WikiData[8] as its target knowledge base. Wikidata's extensive coverage and rich semantic relationships make it an ideal target knowledge base for Entity Linking datasets (Möller, Lehmann & Usbeck (2021)). Many research efforts and evaluation benchmarks in the field of Entity Linking utilize Wikidata as a reference for linking textual mentions to their corresponding entities.

Mewsli-9's corpus genre is different than the Wikipedia text and written in a different style. Therefore, a system that is trained on Wikipedia will be tested in terms of cross-domain/style robustness when evaluated on the Mewsli-9 corpus.

We focused on the Turkish set only for evaluation purposes. In the scope of this work, we conducted experiments with smaller sets of entities due to the limited resources. We trained various models for these sets of entities from Turkish Wikipedia. And for each of these models, only a subset of Mewsli-9 Turkish samples that has entity mentions for the set of entities we are interested in will be used.

We preprocessed the Mewsli-9 Turkish. The main task was mapping Mewsli-9 Turkish entities and Turkish Wikipedia dump entities to each other. This lets us evaluate Turkish Wikipedia-trained models on Mewsli-9 Turkish. After the mapping, we were able to successfully map 5731 out of 5811 samples. The remaining samples could not be mapped due to missing Wikidata IDs on the Turkish Wikipedia side.

In Table 3.2, statistics for the top N entities that have the most in-interwiki in Turkish Wikipedia are shown. The numbers in the table are for the samples in Mewsli-9 that are related to given entity subsets.

Table 3.5 The number of entity mentions, documents, and sentences, also, the average number of entity mentions per document and sentence are described for Mewsli-9 Turkish set. The statistics are calculated for each Top N value as follows; we found the entity set and filter out the rest of the samples which are not referred to our set.

| **Top N** | **#Ent. Ment.** | **#Docs** | **Avg.$_{\mathbf{Doc}}$** | **#Sents.$_{\mathbf{ent.}}$** | **Avg.$_{\mathbf{Sent}}$** |
|---|---|---|---|---|---|
| 100 | 1,014 | 819 | 1.24 | 528 | 1.92 |
| 500 | 1,777 | 1,311 | 1.36 | 729 | 2.44 |
| 1,000 | 2,083 | 1,505 | 1.38 | 789 | 2.64 |
| 2,000 | 2,574 | 1,821 | 1.41 | 853 | 3.02 |
| 5,000 | 3,280 | 2,227 | 1.47 | 902 | 3.64 |
| 10,000 | 3,790 | 2,505 | 1.51 | 937 | 4.04 |
| 50,000 | 5,080 | 3,153 | 1.61 | 984 | 5.16 |
| 100,000 | 5,427 | 3,290 | 1.65 | 990 | 5.48 |
| 564,428 | 5,701 | 3,413 | 1.67 | 994 | 5.74 |

[8] WikiData is a multilingual knowledge graph that is cooperatively revised, and it's hosted by the Wikimedia Foundation.

As it can be seen from Table 3.2 when we take only the top 2,000 or 5,000 entities, we cover more than half of all mentions that exist in Mewsli-9 Turkish, similarly in the Turkish Wikipedia dataset. However, the average entity mention per sentence is higher than the Turkish Wikipedia dataset.

### 3.3 Turkish Tweets for Entity Linking (***TW-SUNLP-EL***)

The majority of research on Entity Linking is focused on datasets that are either Wikipedia-based or formally written text like news (Hoffart, Yosef, Bordino, Fürstenau, Pinkal, Spaniol, Taneva, Thater & Weikum (2011), Cucerzan (2007b), Ratinov, Roth, Downey & Anderson (2011b)). We are introducing a relatively small dataset to test the robustness of transformer-based neural models on a different domain than their training data. The Turkish Wikipedia test set and Mewsli-9 test sets are different domains as well. However, they are both written in formal text. The dataset that we are introducing is based on tweets, therefore, the difference in the domain and text format is bigger.

We inherit a new dataset for Turkish Named Entity Recognition on tweets and labeled them for Entity Linking on top of the annotations for NER. This dataset is called TW-SUNLP and was introduced by Çarık & Yeniterzi (2022). They introduced a new Twitter corpus dataset for the Named Entity Recognition task in Turkish. The dataset comprises 5000 tweets spanning a year and has been meticulously labeled by several annotators, demonstrating a strong consensus. It exhibits diversity in terms of named entity types, encompassing not only individuals, organizations, and locations, but also time, money, product, and TV show categories.
We picked Wikidata as our target knowledge base and have used an open-source tool called INCEpTION, developed by Klie, Bugert, Boullosa, Eckart de Castilho & Gurevych (2018). INCEpTION is an innovative annotation platform that has been specifically developed to support interactive and semantic annotation tasks, which involve activities like concept linking, entity linking, and semantic frame annotation. These tasks are often time-consuming and demanding for annotators, espe-

cially when dealing with complex knowledge bases. To address these challenges, the developers of INCEpTION have incorporated machine learning capabilities into the platform. These capabilities actively assist and guide annotators throughout the annotation process, significantly enhancing efficiency and accuracy. For instance, there were a lot of mentions of Turkey, the country. And during our annotation process, INCEpTION begins to suggest the Wikidata item for Turkey (Q43) when the text **Türkiye** is seen in the given tweet.

To enable annotators to use INCEpTION, we uploaded TW-SUNLP data (5000 samples) with its NER annotation to the INCEpTION system in a pre-annotated format. The examples of uploaded tweets are shown below.

- *Kemal Kılıçdaroğlu* ***PERSON*** *CHP* ***ORGANIZATION*** *'yi tarif etmiş. $URL*
- *'Devlet pazarlık yapmaz, gider alır.' 82 Nijerya* ***LOCATION*** *:)*

Since the named entities are already annotated, the whole annotation process gets faster.

A graduate student annotated the uploaded data for Entity Linking by tagging the annotated named entities with the corresponding Wikidata items. During the annotation, the samples for the annotation process were randomly ordered. The student annotated 3,234 out of 5,000 samples. After a quick look at the annotated data, we found out that the dataset was poorly annotated. There were many named entities that are not linked to Wikidata which actually has a corresponding item. Therefore, a curation phase was needed.

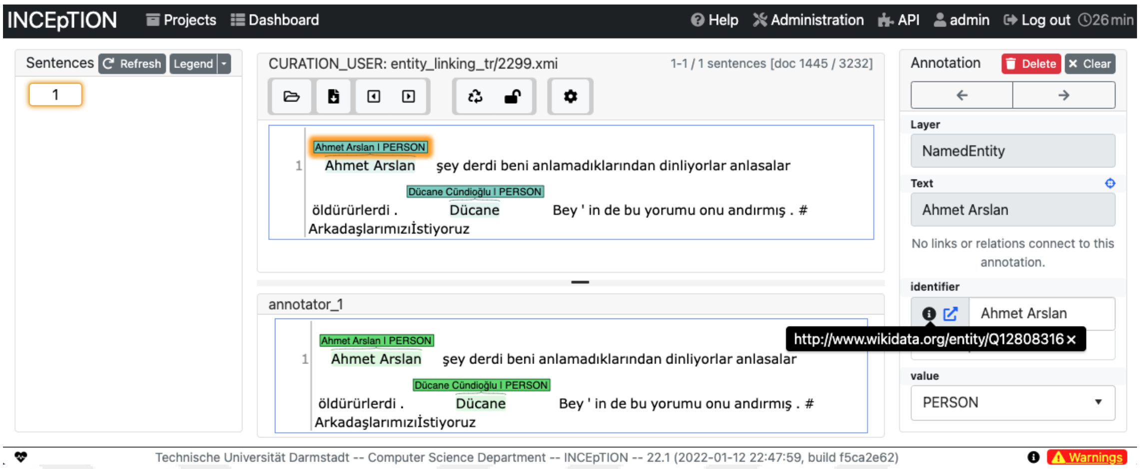
Firstly, we downloaded the annotated data and sorted it in terms of the highest entity mention count for the top 2000 entities in Turkish Wikipedia. Since the annotation was poor, we basically tried to match the named entity text span with the Wikipedia title of entities for the counting process. This lets us build a dataset quicker that we can use for the evaluation of our models. Then, we curated 100 samples starting with the sample which has the most entity mentions that we can use for our evaluation. Figure 3.2 shows the INCEpTION curation page with an example. As it can be seen from Figure 3.2, the curator can see the annotations from each annotator and can merge them into the final version or add/remove annotations.

After the curation, we download the curated data in CONLL[9] format and parse it a format that we can easily use with our models.

We were able to finalize currate 156 tweets. Only 38 of these tweets includes an entity mention for the top 2000 entities of Turkish Wikipedia. Within these 38

[9]The CONLL format is commonly used for representing syntactic dependency trees and is widely adopted in natural language processing tasks, such as named entity recognition, part-of-speech tagging, and parsing. It consists of tab-separated columns representing different linguistic features or annotations for each word in a sentence.

Figure 3.2 The INCEpTION curation page with an example sentence "***Ahmet Arslan*** *şey derdi beni anlamadıklarından dinliyorlar anlasalar öldürürlerdi.* ***Dücane*** *Bey'inde bu yorumu onu andırmış. #Arkadaşlarımızıİstiyoruz*". The Named Entities are mapped to corresponding Wikidata entities.




tweets, we have 158 entity mentions which comes to an average entity mention of 4.15 per tweet.

# 4. TRANSFORMER-BASED ENTITY LINKING MODELS

In this chapter, we delve into Transformer and especially encoder-only Transformer models and their various aspects. We examine the architecture and learning objectives of Transformer models by briefly explaining how they work and what they are good at for learning. Then, we cover Tokenizers which are key components of Transformer models for representing the input sequence. We explain what is their duty for Transformer models. And lastly, we cover a concept called Transfer Learning for Transformer models and mention a few pre-trained transformer models for Turkish. Moreover, we dive into two popular Entity Linking models which are exploiting pre-trained encoder-only Transformer models. The first model is called BLINK and it is using a two-stage approach where both use pre-trained encoder-only Transformer models. It is a state-of-the-art work on Neural Entity Linking that works with a dense-vector retrieval inner mechanism. The second model is called LUKE and it is modifying the original encoder-only Transformer model architecture by transforming the self-attention mechanism into '*entity-aware self-attention*'. We will explain the architecture and the training objectives for these models in further sections.
Finally, we propose a model called EntityBERT that exploits the pre-trained encoder-only transformer models. With this model, we are trying to use the pre-trained encoder-only transformer models for the Entity Linking problem by just manipulating the input data we are using and without changing the architecture.

## 4.1 Background on Transformer Models

### 4.1.1 Introduction

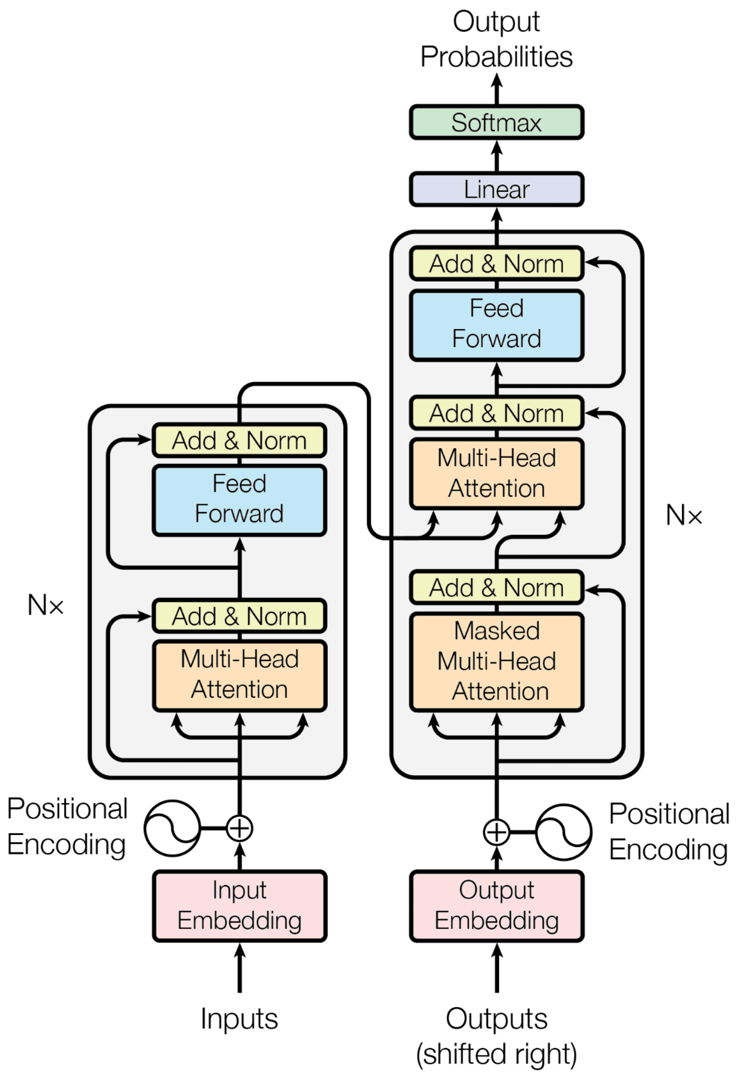
The Transformer architecture is an advanced of deep neural network that uses self-attention mechanisms to process input data (Vaswani et al. (2017)). The Transformer model is originally designed for sequence-to-sequence modelling, therefore is composed of an encoder and a decoder, both of which contain stack of identical layers. The Transformer architecture's central notion is to process the input and output sequences concurrently utilizing a self-attention mechanism. The self-attention method allow the transformer model to balance the attention that is given to all the other tokens and learn a representation for each token based on its relationship with other tokens in the input sequence. The Transformer architecture is made up of an encoder and a decoder. The encoder consists of multiple layers of self-attention and feed-forward neural networks, while the decoder consists of multiple layers of *masked* self-attention and feed-forward neural networks. Figure 4.1 demonstrates the full architecture.

The self-attention mechanism in the original Transformer architecture is applied using three types of embeddings: the query, key, and value vectors. The query and key vectors are used to compute the attention scores between all pairs of tokens in the input sequence, while the value vectors are used to generate the representation vectors for each token.
The multi-head attention mechanism is another crucial component of the Transformer architecture. It involves splitting the query, key, and value vectors into multiple heads and performing the self-attention computation independently on each head. This allows the model to learn different relationships between the tokens at different levels of granularity.
The feed-forward neural networks in the Transformer architecture are used to apply non-linear transformations to the output of the self-attention mechanism.
In summary, the Transformer architecture is a powerful neural network architecture that is based purely on attention mechanisms. It has been shown to achieve state-of-the-art performance on a wide range of natural language processing tasks, including machine translation, language modelling, and text classification.

### 4.1.2 Encoder-only Transformer Models

The original Transformer architecture utilizes an encoder-decoder structure for sequence-to-sequence modelling. In addition to that, there is another variant named 'encoder-only Transformer' models. Encoder-only Transformer models work on encoding input sequences. Text classification, sentiment analysis, and named entity

Figure 4.1 The Tranformer architecture demonstrating the encoder and decoder stacks with their inner mechanism. This diagram is taken from the original paper Vaswani et al. (2017).



identification are just a few of the NLP problems for which this technique has gained popularity and delivered remarkable results.
We can name a couple of examples for the encoder-only Transformer models, such as BERT (Devlin, Chang, Lee & Toutanova (2019)) and RoBERTa (Liu, Ott, Goyal, Du, Joshi, Chen, Levy, Lewis, Zettlemoyer & Stoyanov (2019)). By accurately obtaining contextual information and semantic representations of input text, these works have shown superior performance. In order to understand relationships between words and contextualize their representations, these models use the same key components as the encoder in the encoder-decoder architecture.
For instance, BERT (Bidirectional Encoder Representations from Transformers) is pre-trained utilizing next-sentence prediction (NSP) and masked language modelling (MLM) objectives on large-scale corpora. After pre-training with the aforementioned tasks in an unsupervised manner, the BERT model can be fine-tuned in downstream tasks. The BERT model demonstrated outstanding performance in a variety of downstream tasks, including question answering (QA), named entity recognition (NER), and text classification. Additionally, one other notable encoder-only model is RoBERTa (Robustly Optimized BERT pre-training approach). It builds upon the BERT architecture and training methodology but with additional modifications and optimizations. RoBERTa employs the BERT architecture and removes the NSP task during pre-training. It uses a larger dataset and longer training steps. After the fine-tuning in downstream tasks, RoBERTa brings improvement over the BERT model.

### 4.1.3 Tokenizers

Tokenizer is a crucial component of Transformer models that is used for processing the raw text data. It receives the raw text data and splits it into pieces that is called tokens which can be represented with numerical values. Tokens can be as short as a single byte (Xue, Barua, Constant, Al-Rfou, Narang, Kale, Roberts & Raffel (2022)) or as long as a whole word.
First of all a tokenizer should be trained on the text data that will be used for the training of the Transformer model. During the training, it learns the vocabulary of the Transformer model. Vocabulary consists of words, subwords or characters/bytes. Tokenizer counts the frequencies of each character in the whole text dump and that stays as the starting vocabulary. Then, it starts merging of two items which is the top frequent duo. Each merge creates a new item which is added into the vocabulary with its own frequency. Merging continues until it reaches the vocabulary size limit.

The size of the vocabulary is a hyperparameter set by the user.
After the training, tokenizer can be used as a preprocessing tool before the training and evaluation of the Transformer model by segmenting the raw text and inserting the special tokens of Transformer model. Each item in the vocabulary has a corresponding vector in the embedding layer of the Transformer model.
Pretrained Transformer models has their own pretrained tokenizers as well. One of the advantages of tokenizers is that they can be manipulated. A new text token or a special token can be introduced to the vocabulary and a new entry can take place in the embedding layer. In this work we are exploiting this for representing entities with a single token. This will be explained in detail in the further sections.

### 4.1.4 Transfer Learning with Transformer Models

Transformer models brought a groundbreaking effect to Natural Language Processing with their performance. One other important thing that got popular with Transformer models is a technique called transfer learning. Tranfer learning can be explained in two key steps:

- **Pretraining**: Initially, a model is 'pretrained' on a large dataset and a generic task. The pretrained model captures the contexualized representation of words and 'learns' the language. For example, the generic task can be Masked Language Modelling or Next Word Prediction.
- **Finetuning**: The pretrained weights can be shared with anyone. One can initialize their model with the pretrained weights and 'finetune' their model on a downstream task like Sentiment Analysis, Entity Linking, Question Answering and more.

This process reduces the training time to reach a certain level of performance on the downstream tasks. It can also bring performance improvement and generalizability.

### 4.1.5 Pretrained Transformer Models for Turkish

We have investigated the popular hub for the pre-trained Natural Language Processing models called `huggingface`[1]. There are more than 70 pretrained models that is can perform Masked Language Modelling task for Turkish including large multilingual models.
One of the most important pretrained Transformer model for Turkish is called BERTurk (Schweter (2020)). BERTurk is a BERT model that is trained on Turkish text for Masked Language Modelling and Next Sentence Prediction tasks. It trained on a text corpus with a size of 35 GB. The corpus is consist of a Turkish Wikipedia dump, OSCAR[2] and various other corpora. It was trained on Google Cloud TPUs for more than 2 million steps.
Original BERT model has two variants;

- **large** has a stack of 24 encoder layers with 16 self-attention heads and uses hidden dimension of 1024.
- **base** has a stack of 12 encoder layers with 12 self-attention heads and uses hidden dimension of 768.

The BERT configuration used for BERTurk model is called **base**.

## 4.2 BLINK

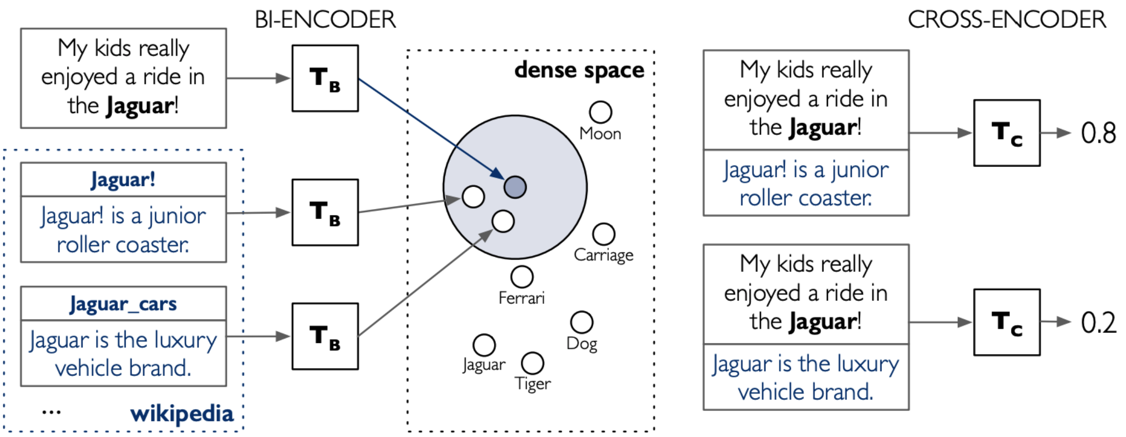
Wu et al. (2020) introduces a model called BLINK which is a conceptually simple, two-stage model for the Entity Linking problem. As in classical Entity Linking systems, there are two steps; candidate generation and re-ranking. These steps use separate models that are based on fine-tuned BERT architectures. In the first stage, they retrieve in a dense space defined by a bi-encoder that independently embeds the mentioned context and the entity description. In the second stage, retrieved top N candidates are re-ranked by a cross-encoder model.

The bi-encoder uses two independent BERT transformer models to encode the entity-mention (and its context) and the entity into dense vectors, and each entity candidate is scored as the dot product of these vectors. The mention and its

[1] huggingface.co

[2] `https://oscar-project.org/`

Figure 4.2 Brief overview of the BLINK entity linking solution: Input is encoded in a shared dense space with entity representations. Nearest neighbors search retrieves $k$ entities, supplied to the cross-encoder. The cross-encoder attends to input text and entity descriptions, producing a probability distribution over the candidates.




context are modelled as follows;

$$[CLS] < left-context > [M_{\mathrm{s}}] < entity-mention > [M_{\mathrm{e}}] < right-context > [SEP]$$

For the BERT model, $[CLS]$ stands for the start of the sentence token and $[SEP]$ stands for the end of the sentence token. The $[M_{\mathrm{s}}]$ and $[M_{\mathrm{e}}]$ tokens are introduced by the authors to identify the start and end of the entity-mention. On the other side, the entity is being modelled as well. The authors structure the input for the entity encoder as follows;

$$[CLS] < entity - title > [ENT] < entity - description > [SEP]$$

The authors introduced the $[ENT]$ token and it's used to separate the title and entity description for the model. The title is the Wikipedia title for the given entity and the entity description is the first sentence of its Wikipedia page.
As the name suggests, the bi-encoder model consists of two pre-trained BERT models. The first BERT model, $T_1$, gets the entity-mention representation, $T_m$, and the second one, $T_2$, gets the entity representation, $T_e$.

$$y_{\mathrm{m}} = cls(T_1(T_{\mathrm{m}}))$$

$$y_{\mathrm{e}} = cls(T_2(T_{\mathrm{e}}))$$

The formulations above show that the outputs of BERT models are being passed through a *cls* function which is basically taking the latent representation of $[CLS]$

token from the outputs of BERT models. Finally, we take the dot product of these representations and obtain a score, $s(m, e)$.
To obtain some negative samples as well, the optimisation happens on the batch-level instead of the sample-level. This means they are calculating scores for each entity-mention in the batch against all the entities in the batch. This is implemented as a multiplication of two matrices where the dims are $b\ x\ H$ and $H\ x\ b$, and the final output is a matrix with a dimension of $b\ x\ b$ where each sample has a corresponding score with each entity in the batch, $s(m_{\mathrm{i}}, e_{\mathrm{j}})$. The bi-encode model tries to maximise the score for the correct entity while minimising the scores for others.
After the training, precomputed and cached entity representations are stored. At inference time, FAISS (Johnson, Douze & Jégou (2019)) is used in order to perform a neighbour search for a given sentence that includes an entity mentioned. FAISS is a software published by Facebook for efficiently applying similarity search and clustering of latent representations which can have relatively big dimensions. Hence, at the inference, the sentence that has the entity mention passes through the entity-mention encoder ($T_1$), and the obtained representation, $y_m$, being used with FAISS to do similarity search in the stored entity-representation space. To improve the overall performance, they are using the second stage which is re-ranking with a cross-encoder model. They concatenated context/mention and entity representations which were mentioned above. They removed the entity representation $[CLS]$ in order to let the cross-encoder learn cross attentions between the two inputs.

$$y_{\mathrm{m,e}} = cls(T_{\mathrm{cross}}(T_{\mathrm{m,e}}))$$

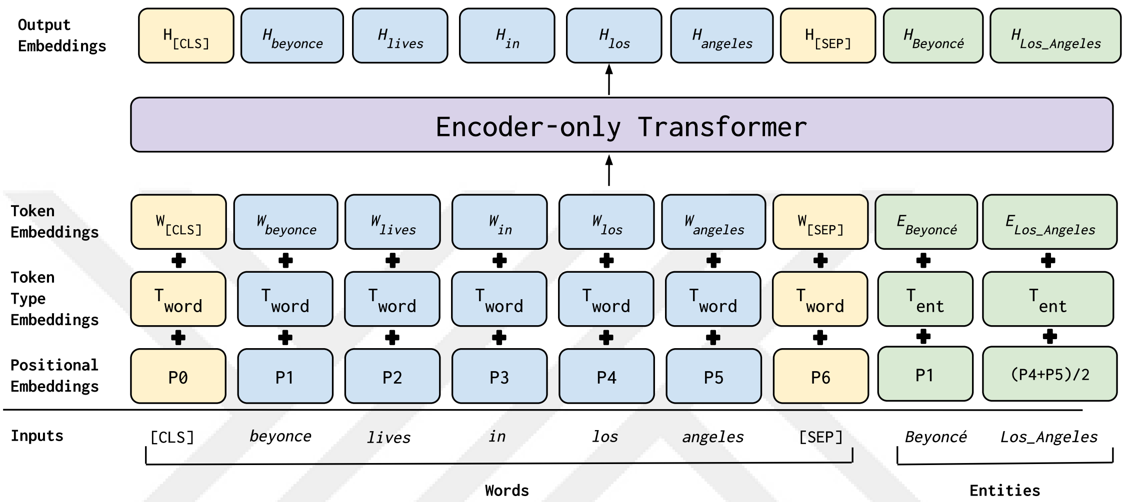
$$s_{\mathrm{cross}}(m, e) = y_{\mathrm{m,e}} W$$

The cross-encoder model, $T_{cross}$, is a single BERT model which receives tokens for both entity-mention and entity representation and outputs a dense representation of given pair, then this representation passes through a linear layer for obtaining a probability, $s_{cross}(m,\ e)$, for how likely the mentioned entity is the one that is concatenated.

## 4.3 LUKE

Yamada, Washio, Shindo & Matsumoto (2022b) introduces an Entity Linking model which exploits contextualised entity embeddings based on an encoder-only Transformer architecture (the BERT model). LUKE (**L**anguage **U**nderstanding with **K**nowledge-based **E**mbeddings) model exploits the Masked Language Modelling (MLM) task to learn contextualized representations for entity tokens. It operates over a knowledge base (KB) which has an infinite number of entities. LUKE model introduces a secondary embedding group for entities in addition to word embeddings. Figure 4.3 illustrates the model architecture, embeddings and input format.

Figure 4.3 Architecture of the LUKE model with the sentence "*Beyoncè lives in Los Angeles.*" with entity annotations on Beyoncè and Los Angeles.




As can be seen from the figure, the input to the encoder-only transformer model is constructed with three different types of embeddings;

- **Positional Embedding** represents the position of a token in the given sequence. It helps transformer models to have an understanding of the orders of tokens. The entity tokens get the positional embedding of the corresponding text token in the sequence. If it is represented by multiple text tokens, then the average of these will be assigned to the entity embedding as its positional embedding. When the masked entity tokens are predicted, this mechanism helps the model to attend the proper location.

- **Token Type Embedding** represents whether the given token is an entity token or text token.

- **Token Embedding** represents the embedding of the corresponding word or entity token. LUKE model uses separate embedding layers for words and entities.

A regular encoder-only transformer model has these three embeddings as well, but each has a slightly different job than what they do in the LUKE model.
The training task for LUKE is called Masked Entity Prediction (MEP), introduced

by the authors, based on the MLM. Similar to the MLM, some entities are randomly masked with a certain probability and the LUKE model tries to predict the masked entities. This can be formulated as follows;

$$\hat{y} = softmax(Bm_{\mathrm{e}} + b_{\mathrm{o}})$$

where $m_{\mathrm{e}}$ stands for dense representation extracted from the encoder-only transformer model. Then it is passed through a linear layer that has an output dimension size that is equal to the number of entities LUKE is being trained for. And finally, there is a softmax layer for transforming the outputs into probabilities. LUKE model is being optimized to maximise the probability value at the index of the target entity and minimise the others.

## 4.4 EntityBERT

We introduce a model called EntityBERT, which is a simpler version of the LUKE model. EntityBERT model tries to learn the contextualized embeddings for entities within the entity-mention sentences by replacing the text tokens with entity tokens. Figure 4.4 shows the architecture and input structure in detail.

We introduce a set of new tokens, $V_{\mathrm{E}}$, to the current vocabulary, $V_{\mathrm{W}}$, and to the embedding layer where each token represents an entity. With this addition, the vocabulary size increases to $V_{\mathrm{A}} = V_{\mathrm{W}} + V_{\mathrm{E}}$. The replacement is a probabilistic event. With respect to certain probability we set, the entity mention spans can be replaced with the special entity token or they can stay as they are. Figure 4.5 demonstrates an example.

We apply the replacement according to probability to make sure that our model does not forget about the text tokens being used to represent entities. With this model, we aimed to simplify the LUKE model by solving Masked Entity Prediction and Masked Language Modelling problems together in one task.

Figure 4.4 Architecture of the EntityBERT model with the sentence "*Beyoncè lives in Los Angeles.*" with entity annotations on Beyoncè and Los Angeles. The $ENT\#2243$ token is the entity token that is corresponding to Beyoncè in our knowledge base.

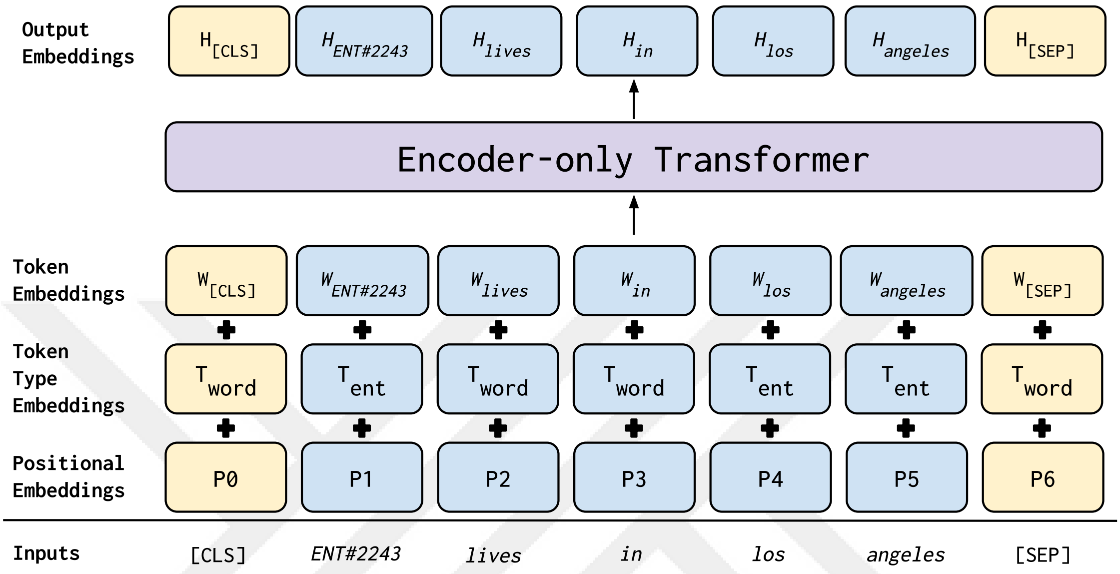
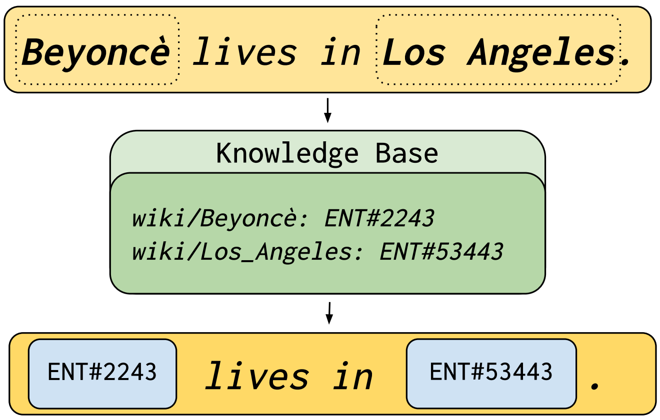



Figure 4.5 An example for entity replacement during training with the sentence "*Beyoncè lives in Los Angeles.*". This replacement occurs with a probability that is set by us as a hyperparameter, similar to masking probability for the MLM task.



# 5. EXPERIMENTS AND RESULTS

In this chapter, we will explain the details of our experiments and relevant scores or analyses along with the experiment. Additional to our EntityBERT experiments, we have trained two other popular models for Entity Linking on our Turkish dataset. These models are called BLINK introduced by Wu et al. (2020) and LUKE introduced by Yamada et al. (2022b). The details on how these models work and our experiments with these models on Turkish data will be explained below.
Before getting into the details of the experiments we have conducted, there are a few things that are common among these experiments to be explained for not to step into repetition. Due to our limited resources (8 and 24 GB GPUs in two different machines), a pre-training of a transformer model from scratch with a full Turkish Wikipedia was impractical to achieve. Therefore, we have conducted experiments based on a pre-trained Turkish transformer. In some of those fine-tuning experiments, we initialized some components from scratch while still using most of the pre-trained components. Moreover, we have introduced only a limited amount of Entity tokens to our tokenizer and model for fine-tuning. We have parsed the Turkish Wikipedia and extracted the frequencies of each entity. Then, we only used the most frequent 2000 entities for conducting our experiments.

## 5.1 Evaluation Metric

Although we consider the Entity Linking problem to be a Natural Language Processing task, it also falls within the category of information retrieval. This is relevant when we consider the proper evaluation metric. There few evaluation metrics used for information retrieval tasks, namely Precision@K, Recall@K, and Accuracy@K. These metrics are similar to Precision, Recall, and Accuracy, however, there is a difference. In some information retrieval tasks, there are multiple ‘correct answers’ (targets). Therefore, the models that we are using for these tasks should return a list of ‘candidates’.

$$Pre@K = \left( \sum_{i=1}^{N} \frac{\#\text{ correct predictions within the top K predictions}}{K} \right) / N \tag{5.1}$$

$$Rec@K = \left( \sum_{i=1}^{N} \frac{\#\text{ correct predictions within the top K predictions}}{\#\text{ gold targets}} \right) / N \tag{5.2}$$

For the sample set of size N, Precision@K and Recall@K metrics are formulated as in Equation 5.1 and Equation 5.1, respectively. The Accuracy@K metric is the same as Recall@K for this setting.
Our task, Entity Linking with Wikipedia, is slightly different than what we have explained above. For our task, we only have a single correct answer. This means all of these metrics above can take a value of 0 or 1 for a single sample, therefore, they mean the same thing. Hence, we moved forward with the Accuracy@K metric for different K values. For instance, for an entity mention of *Amerika Birleşik Devletleri*, if our model returns a list of candidates where $ENT\#234$ sits in the 3rd place, and if we compute Accuracy@1, Accuracy@3, and Accuracy@10 metrics; then the scores would be 0, 1 and 1 respectively.

## 5.2 BLINK Experiments

BLINK model uses a pre-trained model to initialize encoders in its bi-encoder and cross-encoder modules. We have modified the code slightly to use the pre-trained `huggingface`[1] models for the initialization of encoders. We have initialized the encoder modules with the `dbmdz/bert-base-turkish-uncased`[2] model. Considering the architecture of the bi-encoder model, two of the same are being trained together.

[1] https://huggingface.co/

[2] https://huggingface.co/dbmdz/bert-base-turkish-uncased

The first one learns the representation of entity mentions. The second one learns representation for the entities and implicitly the entity space. We trained the bi-encoder model with a learning rate of `1e-05`, as the original paper suggests. We used the maximum length of 32 tokens for context and entity representations. This means for an entity mention span with $a$ tokens, there will be only $(32-a)/2$ tokens on each side (left and right) of the entity mention.

### 5.2.1 Turkish Wikipedia Top 2000 Entities

As it was mentioned several times, we had access to limited resources. Therefore, we trained our BLINK model on a subset of Turkish Wikipedia. We obtained the top 2000 frequent entities (highest in-interwiki count) and all their mentions from the Turkish Wikipedia dump and we processed it to the format BLINK expects. Below, you can see how each sample is represented before feeding into BLINK scripts. Then, trained the BLINK bi-encoder model with our dataset to learn an entity space of 2000 entities. We trained all the layers bi-encoder with a batch-size of 16 for 1 epoch on two 8 GB GPUs.
Since the entity space is quite small, the bi-encoder model achieved impressive results on the test of Turkish Wikipedia. However, to have the full experiment, we trained the cross-encoder model.
BLINK's cross-encoder model requires precomputed candidates. Therefore a full inference with the bi-encoder model is needed for the training set. We ran a full inference and saved the top 8 candidates predicted by the bi-encoder for each sample. We trained the cross-encoder with the same pre-trained model that we used for the bi-encoder, BERTurk. We trained all layers of the model. Following the similar steps of the authors, we limited the training samples. The original BLINK model used only 1 million training samples for the training of the cross-encoder. We used only 250,000 samples. The cross-encoder model was trained with a learning rate of 1e-4 and we were only able to fit a batch size of 4 to 2 of 8 GB GPUs.
The training of the bi-encoder took around 31 hours and the training of the cross-encoder took almost 10 hours.
We evaluated the models in three different datasets that we explained in Chapter 3. All of these datasets only contain samples that have one or more entity mentions for an entity that is in the top 2000 entities.

Table 5.1 shows the results for the Turkish Wikipedia test set we have created. This is the biggest evaluation set by a margin. It contains samples that are in the

Table 5.1 BLINK results on Turkish Wikipedia test set.

| **Model** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** | **Acc@64** |
|---|---|---|---|---|---|---|
| $\text{BLINK}_{\text{bi-encoder}}$ | 95.86 | 99.62 | 99.77 | 99.84 | 99.89 | 99.96 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 96.22 | 99.66 | 99.77 | 99.90 | 99.93 | 99.96 |

Table 5.2 BLINK results on Mewsli-9 Turkish set.

| **Model** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** | **Acc@64** |
|---|---|---|---|---|---|---|
| $\text{BLINK}_{\text{bi-encoder}}$ | 96.38 | 99.96 | 99.96 | 100.00 | 100.00 | 100.00 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 97.12 | 99.96 | 99.96 | 100.00 | 100.00 | 100.00 |

same domain and genre as the training set. $\text{BLINK}_{\text{bi-encoder}}$ performs strongly even though the same set is quite big. The cross-encoder was not able to bring a big improvement since the room for improvement was quite small.

Table 5.2 shows the scores obtained by the BLINK model for the Mewsli-9 set. The $\text{BLINK}_{\text{bi-encoder}}$ model is already achieving extremely good results therefore there is not much to improve for the cross-encoder model, similar to the Turkish Wikipedia test set. Also, it has been shown that the BLINK model is robust to 'genre' change in the text when experimenting with a small set of entities. As it was mentioned before, while the training set for the BLINK model was using Wikipedia Articles only, the Mewsli-9 set consist of news articles.

Table 5.3 reports the results of the BLINK model for the TW-SUNLP-EL dataset that we have annotated. This set was the most challenging set for the BLINK model. The main reason could be the following. Tweets are quite short to provide a good context for the entity mentions. Also, the text we are dealing with in this set is informal and there are misspellings/typos.

### 5.2.2 Turkish Wikipedia All Entities

Table 5.3 BLINK results on TW-SUNLP-EL set.

| **Model** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** | **Acc@64** |
|---|---|---|---|---|---|---|
| $\text{BLINK}_{\text{bi-encoder}}$ | 85.53 | 98.03 | 98.03 | 98.03 | 99.34 | 99.70 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 87.26 | 98.67 | 98.67 | 98.93 | 99.34 | 99.70 |

At some point during our work, we had access to 8 GPUs of 32GB for a short period of time. Within this period, we were able to train a BLINK bi-encoder model. Since the resources were allowing, we trained this model with the all Turkish Wikipedia of that time.

The training setting was a bit different. The BLINK model works on a latent space of entities where representations are extracted by feeding the entity description into the BLINK encoder. This setting allows the users to work in the zero-shot Entity Linking setting. In other words, the entities in the test set they created from the English Wikipedia dump do not have any samples in the training set. They first separated a set of entities for their training, validation, and test set. Then collected all the entity mentions for the entities for each set. Therefore, when they evaluate their test set, each sample's target entity was not seen by the BLINK model during the training.

We implemented the same data processing step for Turkish Wikipedia. The Wikipedia dump parser that we used for this operation was not as robust as the latest one we developed, therefore the parsing was not as fruitful as it could be. Table 5.4 shows the statistics for the dataset we used.

Table 5.4 The dataset statistics for train, validation, and test set that we created from the Turkish Wikipedia dump. The number of unique entities (**#UniqueEntity (UM)**), entity mentions (**#EntityMention (EM)**), and the average number of mentions per entity (**EM/UM (Avg)**) is shown.

| **Set** | **#UniqueEntity (UM)** | **#EntityMention (EM)** | **EM / UM (Avg)** |
|---|---|---|---|
| Train | 167,558 | 1,442,109 | 8.60 |
| Validation | 2,670 | 19,177 | 7.18 |
| Test | 8,669 | 65,729 | 7.58 |
| All | 178,897 | 1,527,015 | 8.53 |

We trained the bi-encoder with a learning rate of 1e-5. Since we had bigger GPUs, we could fit a batch size of 128. Additionally, we were using up to 128 tokens for representing the entity descriptions.

We evaluated this model on two sets that were available at that period. Table 5.5 shows the scores for the Turkish Wikipedia test set and Mewsli-9 Turkish set. It can be seen that the BLINK model is still robust for a genre change in the text even with a much larger entity set when we compare the scores for the Turkish Wikipedia test set and Mewsli-9 Turkish set. Moreover, even though Acc@1 does not seem as promising as we have seen with the top 2000 entities setting, the BLINK model can still achieve almost 92% accuracy in the Acc@4 metric. And of course, it should be kept in mind that these results are for zero-shot settings.

Table 5.5 The scores obtained with BLINK bi-encoder model on Turkish Wikipedia test set and Mewsli-9 Turkish set. This model is trained over all entities in Turkish Wikipedia.

| **Set** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** |
|---|---|---|---|---|
| Test | 77.08 | 91.48 | 95.41 | 97.36 |
| Mewsli-9 | 76.14 | 91.94 | 95.11 | 96.84 |

### 5.2.3 Pretrained English BLINK on Tweets

Our investigation with BLINK in Turkish showed that the BLINK model is quite strong and robust in Turkish. The only challenging set was the TW-SUNLP-EL set since it consists of tweets. However, we used the top 2000 entities for this training and again annotated tweets for TW-SUNLP-EL which has the highest number of mentions for the same set of top 2000 entities. This means the BLINK model was exposed to many samples for each of the entities we are testing it with. To extend the evaluation, we did a small experiment in English.

The BLINK model's code and the pre-trained models in English are available publicly. The authors evaluated their model in some of the popular Entity Linking evaluation sets for English. We wanted to take this one step further and evaluate their English models on available English Tweets that are annotated for Entity Linking.

In one of the recent papers on Entity Linking with tweets, Harandizadeh & Singh (2020), they proposed their own model called Tweeki and bench-marked their model on multiple different evaluation sets for Entity Linking in English Tweets. We were able to find only two of them below.

- **NEEL2016** was introduced by Rizzo, van Erp, Plu & Troncy (2015).
- **TweekiGold** was introduced by Harandizadeh & Singh (2020).

The BLINK model released by the authors scored 49.6 and 79.4 for NEEL2016 and TweekiGold sets, respectively. We were unable to investigate the big performance difference between these two different sets.

## 5.3 LUKE Experiments

The LUKE is a transformer model that utilizes the core concepts of encoder-only transformer models for learning contextualized representations for entities. It exploits the Masked Language Modelling-based pretraining task for predicting entities that are mentioned at some span in the given text. Its scripts and training configurations are publicly available[3].

By its design, it can perform multiple downstream tasks like Question Answering, Named Entity Recognition, and more. Pretrained models for these tasks are also publicly available at `huggingface`[4]. However, Entity Linking models are not available there. After reaching to the authors, we got pretraining instructions for the Entity Linking task[5].

The LUKE model for Entity Linking is utilizing a pretrained encoder-only transformer model. While the BLINK model uses pretrained networks as its component for a bigger pipeline, LUKE is using a modified version of an encoder-only transformer model as it was explained in Section 4.3. It uses the pretrained network to initialize the common components like the encoder stack and embeddings for the tokens.

LUKE has its own data processing pipeline where the input is the Wikipedia dump itself. This pipeline parses the dump and saves the necessary datasets to the disk for further steps (training). Therefore, it is not trivial to intervene with this pipeline for manipulating to have the same splits we use for BLINK and EntityBERT models. However, we were able to force the model to use the top 2000 entities only.

Table 5.6 LUKE local and global inference results on Turkish Wikipedia test set.

| **Model** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** | **Acc@64** |
|---|---|---|---|---|---|---|
| $\text{LUKE}_{\text{local}}$ | 91.00 | 99.76 | 99.86 | 99.86 | 99.90 | 99.95 |
| $\text{LUKE}_{\text{global}}$ | 91.46 | 97.90 | 98.18 | 98.42 | 98.84 | 99.08 |

After building the training dataset out of the Turkish Wikipedia dump, we followed

Table 5.7 LUKE local and global inference results on Mewsli-9 Turkish set.

| **Model** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** | **Acc@64** |
|---|---|---|---|---|---|---|
| $\text{LUKE}_{\text{local}}$ | 91.45 | 97.90 | 98.18 | 98.42 | 98.84 | 99.08 |
| $\text{LUKE}_{\text{global}}$ | 91.14 | 99.76 | 99.86 | 99.86 | 99.90 | 99.95 |

[3] `https://github.com/studio-ousia/luke/tree/master`

[4] `https://huggingface.co/docs/transformers/model_doc/luke`

[5] `https://github.com/studio-ousia/luke/tree/master/examples/entity_disambiguation`

Table 5.8 LUKE local and global inference results on TW-SUNLP-EL set.

| Model | Acc@1 | Acc@4 | Acc@8 | Acc@16 | Acc@32 | Acc@64 |
|---|---|---|---|---|---|---|
| $\text{LUKE}_{\text{local}}$ | 81.48 | 98.14 | 98.14 | 98.14 | 98.14 | 98.14 |
| $\text{LUKE}_{\text{global}}$ | 81.48 | 98.14 | 98.14 | 98.14 | 98.14 | 98.14 |

the instructions and trained the Turkish LUKE model in two stages. For both stages, we used 2 GPUs of 8GB.

In the first stage, we used the BERTurk model for initializing the model. We used a masking probability of 0.0 for regular text tokens and 0.3 for entity tokens. The weights of the pretrained model were fixed and we trained the model for 1 epoch using a batch size of 128 and a learning rate of 5e-4. It uses `AdamW` (Loshchilov & Hutter (2019)) optimizer that uses betas values of $[0.9, 0.999]$, and epsilion of 1e-6. Finally, the weight decay was set to 0.01. The first stage of training is designed to warm up the model by training it for an epoch to learn entity embeddings by fixing the pretrained model components. It took 12 hours of training.

In the second stage of the training, the main difference is that we did not freeze the BERTurk weights. Therefore, the number of activations to keep in the memory during training increased drastically and we were able to fit only a batch size of 8. Also, we decreased the learning rate to 1e-5 and trained for 3 epochs. The second stage of the training took around 30 hours.

LUKE model supports two types of inferences; local and global. The local inference predicts each entity mention independently from other entity mentions by just exposing the model to the entity mention sentence. The global inference is designed to work at the document level for the original LUKE implementation. However, since we are working with a subset of entities, we are not working at the document level. In the original implementation of LUKE global inference, the model predicts entities autoregressive manner. It starts with predicting the entity that occurs first and fills its mask with its prediction, then feeds it into the model for the next entity prediction. According to their reported results, global inference brings improvement over local inference.

The results we obtained with the LUKE model's local and global inferences over the Turkish Wikipedia test, Mewsli-9 Turkish, and TW-SUNLP-EL sets can be seen in Table 5.6, Table 5.7 and Table 5.8 respectively. It can be seen that global inference does not bring improvements over local inference for Mewsli-9 and TW-SUNLP-EL sets in the Acc@1 metric. However, it brings improvement for Acc@4 and bigger metrics in the Mewsli-9 set. In the Turkish Wikipedia test set, global inference brings improvement in the Acc@1 metric but worsens the others. Also, similar to BLINK, the overall performance decreases when it's exposed to the TW-SUNLP-EL set which is in a different domain than its training data.

## 5.4 EntityBERT Experiments

In this section, we will examine the experiments we conducted with our proposed method. Firstly, we will explain three different inference types we tested. Then, we explain how we utilize the tokenizer of the pretrained model for our purpose. Finally, we will delve into the experiments we have conducted where each is about a different component or hyperparameter of our model. Each experiment will be discussed with its results, if available, to validate its effect.

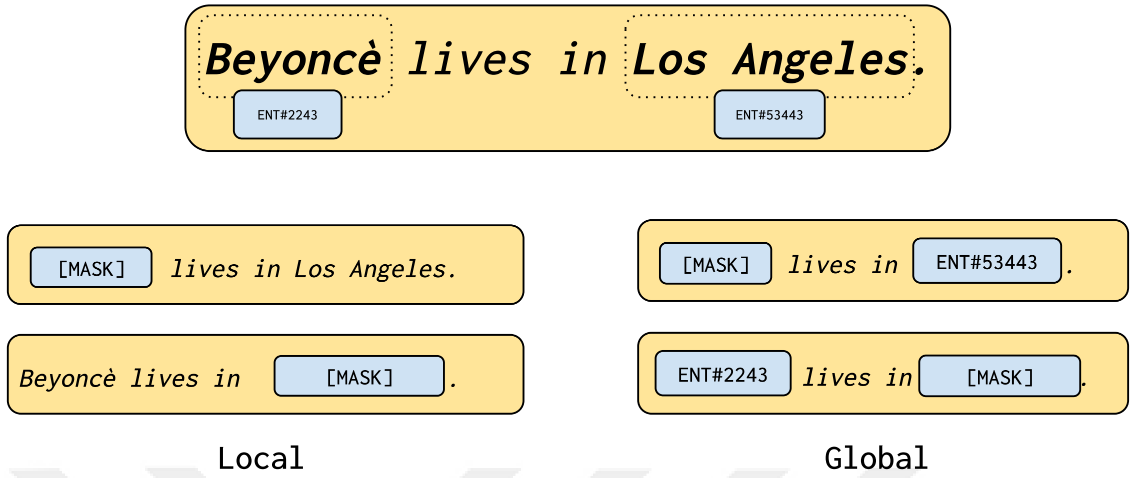
### 5.4.1 Inference Types

There are two common inference types for the Entity Linking task. These types differentiate on the model's exposure to different contexts during the inference. We tried two different inference methods which will be explained shortly. Additionally, as it is explained above, our model has a single vocabulary where regular text tokens and entity tokens are mixed. During the inference, it produces a probability distribution over the whole vocabulary for every masked token. We only use the probability distribution over the entity tokens since we are trying to solve the Entity Linking problem in a non-end-to-end fashion. However, the metrics show that even if we do not do this, regular text tokens are not interfering with EntityBERT's predictions for entity tokens.

- **local** inference exposes our model to the entity mention span and its context only in a given sentence. If there are other entities, we keep the raw text for these entities and treat them as regular text tokens.
- **global** inference exposes our model to the entity mention span, its context, and other entities if there are any. For a sentence with multiple entity mention, we mask one of them and replace others with their entity token iteratively during prediction.

Figure 5.1 demonstrates how each of these inference types is formatting the input

Figure 5.1 Examples of data format for inference types with the sentence "*Beyoncè lives in Los Angeles.*". While **parallel** inference keeps the number of samples to be inferred the same, others need a sample per entity mention.




data with a sample sentence of two entity mentions.

### 5.4.2 Tokenizer

The EntityBERT model uses only a single pre-trained model. We have used the BERTurk model (Schweter (2020)), `dbmdz/bert-base-turkish-uncased`, that we have used for the BLINK model. The encoder, language model head (LMHead), and embedding layer weights are initialized with the pre-trained model we have provided.

One of the hyperparameters we have for the EntityBERT model is the number of entities we want to use during the training. We set a number for this hyperparameter, $N$, the data module we have created will gather the top frequent $N$ entities, then iterate over each sentence in the whole Turkish Wikipedia that has a mention for one of these entities. Then, these mentions will be split into train, validation, and test sets on-the-fly just before training starts.

The tokenizer of the BERTurk model has a vocabulary size of 32000. Each of the entities at the top frequent $N$ list is treated as a single token. For instance, the Turkish Wikipedia page for the United States of America (*Amerika Birleşik Devletleri* in Turkish) is transformed into a code; $ENT\#234$. And each of these entity codes is added to the tokenizer vocabulary. Therefore, the final size of the vocabulary is $32000 + N$. The BERT models have an embedding layer with a weight matrix of *hidden_dim x vocabulary_size*. The BERTurk that we have used is in the configuration of `base` model where a token is represented with a vector of size 768. Therefore the embedding layer of the BERTurk model initially was $32000x768$. After adding the entity codes to the tokenizer, we also resized the embedding layer to have corresponding embeddings for the new tokens we introduced. The final embedding layer size was $(32000+N)x768$. For most of the experiments, we used 2000 as the number of entities we wanted to include.

## 5.5 Data Collator Function

We modified the collator function for our `DataLoaders`. The collator function for the `DataLoader` is the function that applies the needed transformation or processing just before samples are exposed to the model within a batch. Our collator function applies three things according to the three hyperparameters we provide;

- *ent_id_prob* has a probability value that will be used for each entity occurrence in the given sample for whether we would like to keep the raw entity text tokens or replace them with entity code. This will help our model to learn the representation for entity code tokens and also not forget the raw text tokens that are used for the given entity. Once this is done, we have text and entity tokens in the given sample.

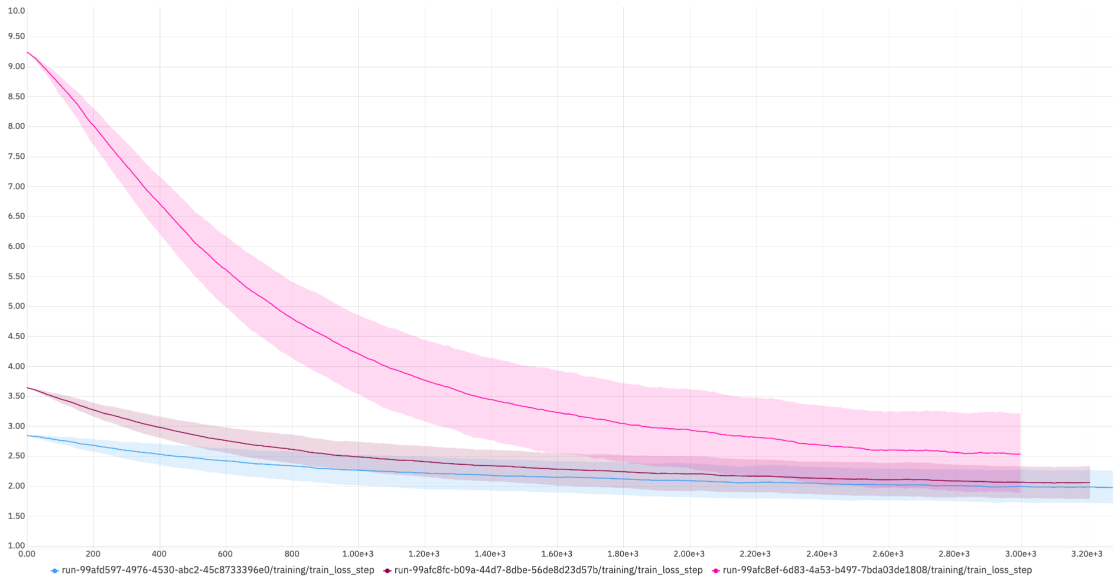
- *ent_mask_prob* has a probability value that will be used for applying masking for the entity code tokens. In other words, replacing the entity code token with the mask token, $[MASK]$.

- *text_mask_prob* has the same functionality with `ent_mask_prob` but for regular text tokens.

One of the important questions here was whether we should expose our model to the raw entity tokens, i.e. we can replace every occurrence of *Amerika Birleşik Devletleri* with $ENT\#234$ and not expose our model to the tokens *['Amerika', 'Birleşik', 'Devletleri']* which basically means setting the *ent_id_prob* to 1.0.
The other important question was, whether we should keep training the pretrained models for filling the mask of regular text tokens. We experimented with different values for the aforementioned hyperparameters. After trying several combinations, we set the value of 0.8, 0.5, and 0.3 for *ent_id_prob*, *ent_mask_prob*, and

*text_mask_prob* respectively. To conduct this hyperparameter search, we manually set a few combinations for these probabilities and ran quick trainings to inspect the loss and metrics. We picked three different combinations for probabilities to demonstrate the effect of these hyperparameters on the model convergence (training-loss) and model performance (evaluation metrics). In Figure 5.2, training loss for three different trainings is demonstrated. The probability combinations were (1.0, 0.5, 0.0), (0.8, 0.5, 0.0), and (0.8, 0.5, 0.3) for `ent_id_prob`, `ent_mask_prob`, and `text_mask_prob` respectively. These losses show that exposing our model to entity

Figure 5.2 Training loss for three different trainings of EntityBERT model. Pink (—), brown (—) and blue (—) lines correspond to (1.0, 0.5, 0.0), (0.8, 0.5, 0.0), and (0.8, 0.5, 0.3) probabilities for `ent_id_prob`, `ent_mask_prob`, and `text_mask_prob` respectively.




raw tokens and also letting it fill mask for regular text tokens is making the model convergence faster and better in terms of loss value. However, this is not enough to conclude. Table 5.9 shows the scores these values as well.

As it can be seen in Table 5.9, exposing our model to entity raw text improves the

Table 5.9 The scores obtained with EntityBERT model on Turkish Wikipedia test set for different probability combinations used by the collator function. This model is trained over top 2000 entities in Turkish Wikipedia.

| **Prob. Comb.** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** | **Acc@64** |
|---|---|---|---|---|---|---|
| (1.0, 0.5, 0.0) | 37.10 | 57.70 | 64.80 | 76.40 | 85.70 | 88.30 |
| (0.8, 0.5, 0.0) | 38.30 | 58.80 | 66.50 | 75.60 | 84.10 | 88.30 |
| (0.8, 0.5, 0.3) | 39.80 | 59.70 | 69.40 | 78.60 | 86.60 | 90.50 |

scores from 37.10 to 38.30. On top of that, if we keep the regular text token predictions during the training, the scores improve by around 1.5 points in Accuracy@1 score.

### 5.5.1 Token Type Embeddings for Entities and Text Tokens

In this experiment, we are exploiting the `token_type_ids` feature of transformer models. `token_type_ids` were used for the original BERT model, to help the model differentiate between different sentences while performing the next sentence prediction task. It has only two embeddings; one for the first sentence and the other one for the second sentence, if available. In our experiments, we are not training or evaluating our models on any task that needs sentence-pair or next-sentence prediction. This means only the first embedding is being used.

During the fine-tuning, we reinitialized the second embedding for `token_type_ids`.

Table 5.10 Token Type Embeddings effect shown with evaluation metrics in **Global** inference setting. **Ent. Only** and **All** columns are the scores we obtain when we consider only the entity tokens and all tokens in the predicted logits, respectively.

| | | **Ent. Only** | | | | | **All** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Set** | **TT** | **@1** | **@4** | **@8** | **@16** | **@32** | **@1** | **@4** | **@8** | **@16** | **@32** |
| TW | Yes | 28.90 | 55.30 | 65.10 | 73.00 | 78.30 | 28.90 | 55.30 | 65.10 | 73.00 | 78.30 |
| | No | 27.60 | 48.00 | 58.60 | 67.10 | 76.30 | 7.90 | 32.20 | 38.80 | 50.00 | 60.50 |
| M9 | Yes | 44.20 | 66.10 | 73.20 | 83.10 | 85.40 | 42.70 | 63.20 | 72.90 | 81.20 | 81.40 |
| | No | 43.20 | 66.00 | 72.40 | 81.70 | 84.50 | 41.90 | 63.10 | 72.10 | 80.20 | 81.40 |
| Test | Yes | 46.50 | 65.10 | 73.50 | 81.50 | 88.40 | 43.00 | 62.10 | 70.90 | 79.20 | 86.40 |
| | No | 47.20 | 66.00 | 74.20 | 82.00 | 88.70 | 47.20 | 66.00 | 74.20 | 82.00 | 88.70 |

While the first embedding was used for regular text tokens, the second one was used for entity tokens (both for entity raw text tokens and entity code tokens). Since all text tokens and embedding tokens are in the same embedding layer and vocabulary, we are using `token_type_ids` as a helper for our model to differentiate entity tokens and text tokens. The results of our evaluations for the effect of Token Type Embeddings usage are presented in two tables. Both tables reports Accuracy@K scores obtained by two variants of EntityBERT models; one utilizes (Yes) and the other one does not utilize (No) Token Type Embeddings (TT) on our three evaluation sets TW-SUNLP-EL (TW), Mewsli-9 Turkish set (M9), and Turkish Wikipedia test set. While Table 5.10 reports scores with **Global** inference setting, Table 5.12 reports for the **Local** setting. Also, each table has scores from **Acc@1** to **Acc@64** for both **Ent. Only** and **All** settings.

Table 5.10 and 5.12 both show that overall, using Token Type Embeddings as a separator for entity and regular text tokens improves the results. One interesting insight we can see is that for both Global and Local settings when we evaluate our models on TW-SUNLP-EL set without using Token Type Embeddings, our model is being confused with the regular text tokens. This can be seen when we look at the scores for **No** and compare **Ent. Only** and **All**. The difference is greatest for **Yes** and **No** in TW-SUNLP-EL set overall. However, when we look at **Acc@1** for **Local** setting and **Ent. Only** section, **No** performs slightly better.

Table 5.11 Token Type Embeddings effect shown with evaluation metrics in **Local** inference setting. **Ent. Only** and **All** columns are the scores we obtain when we consider only the entity tokens and all tokens in the predicted logits, respectively.

| | | **Ent. Only** | | | | | **All** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Set** | **TT** | **@1** | **@4** | **@8** | **@16** | **@32** | **@1** | **@4** | **@8** | **@16** | **@32** |
| TW | Yes | 27.60 | 55.30 | 63.20 | 71.70 | 79.60 | 27.60 | 55.30 | 63.20 | 71.10 | 79.60 |
| | No | 29.60 | 50.70 | 55.90 | 65.80 | 71.10 | 7.90 | 31.60 | 40.10 | 46.70 | 55.90 |
| M9 | Yes | 44.20 | 66.10 | 73.20 | 83.10 | 85.40 | 42.70 | 63.20 | 72.90 | 81.20 | 81.40 |
| | No | 43.20 | 66.00 | 72.40 | 81.70 | 84.50 | 41.90 | 63.10 | 72.10 | 80.20 | 81.40 |
| Test | Yes | 47.40 | 66.00 | 74.20 | 82.00 | 88.70 | 47.20 | 66.00 | 74.20 | 82.00 | 88.70 |
| | No | 46.40 | 65.10 | 73.50 | 81.50 | 88.40 | 43.20 | 62.20 | 71.00 | 79.30 | 86.50 |

Table 5.12 Token Type Embeddings effect shown with evaluation metrics in **Local** inference setting. **Ent. Only** and **All** columns are the scores we obtain when we consider only the entity tokens and all tokens in the predicted logits, respectively.

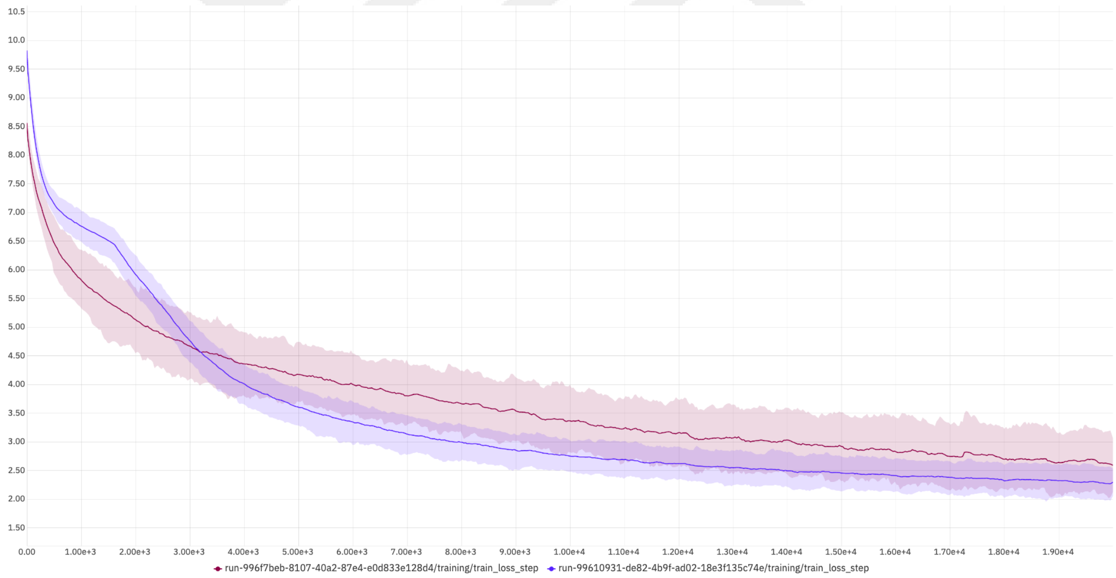
| **Set** | **TT** | **@1** | **@4** | **@8** | **@16** | **@32** |
|---|---|---|---|---|---|---|
| TW | Yes | 27.60 | 55.30 | 63.20 | 71.70 | 79.60 |
| | No | 29.60 | 50.70 | 55.90 | 65.80 | 71.10 |
| M9 | Yes | 44.20 | 66.10 | 73.20 | 83.10 | 85.40 |
| | No | 43.20 | 66.00 | 72.40 | 81.70 | 84.50 |
| Test | Yes | 47.40 | 66.00 | 74.20 | 82.00 | 88.70 |
| | No | 46.40 | 65.10 | 73.50 | 81.50 | 88.40 |

### 5.5.2 Reinitalizing the LM Head of the Pre-trained Model

At a high level, we can think of the transformer models we used as two-component models; encoder and MLM head. MLM head receives the last latent representation of tokens from the encoder and makes a prediction for the masked tokens. When we

initialize our model, both the encoder and MLM head weights are initialized from the pre-trained model. While the encoder part is for learning the contextualized representation of tokens, the LM head makes the prediction for the masked tokens. Since our model is pretrained, LM Head is trained to do predictions for regular text tokens. We wanted to examine the performance of our model with an LM head that is initialized from scratch. Therefore, we applied this idea and only loaded the encoder module of BERTurk and create the LM Head from scratch. Our hypothesis was that; reinitializing the LM Head makes our model more flexible in terms of learning capabilities.
Figure 5.3 shows the loss curves for two variants of EntityBERT. The first model is with a pretrained LM head (—) and the second one (—) is with an LM head initialized from scratch.

Figure 5.3 The loss curves for two training of EntityBERT model. Purple line (—) is showing the loss for an EntityBERT model initialized with BERTurk model encoder and an LM head from scratch. Brown line (—) is showing the loss for an EntityBERT model fully initialized with BERTurk.




As it is expected, the one with the pretrained LM head starts with a better loss however the other one gets better in terms of loss value. However, the evaluation metrics give more interesting insights. Table 5.13 shows the Accuracy@K scores for EntityBERT models with pretrained LM Head and initialized-from-scratch LM head.

When we look at the scores, we see that initializing the LM Head from scratch brings

Table 5.13 Accuracy@K scores for EntityBERT model on TW-SUNLP-EL (**TW**) and Turkish Wikipedia test set (**Test**).

| **Set** | **LM Head** | **Acc@1** | **Acc@4** | **Acc@8** | **Acc@16** | **Acc@32** |
|---|---|---|---|---|---|---|
| TW | Pretrained | 18.10 | 28.50 | 43.70 | 55.50 | 67.10 |
| | Initialized | 26.10 | 35.50 | 50.70 | 62.50 | 69.10 |
| Test | Pretrained | 37.40 | 56.10 | 64.20 | 76.30 | 84.70 |
| | Initialized | 40.00 | 59.20 | 68.40 | 77.50 | 85.60 |

improvement for both datasets. However, the interesting part is that; in the model that uses an LM Head from scratch (**No**) the difference in **Acc@1** score is around 14 points across datasets. However, when we do the same comparison for the variant that uses a pretrained LM head, the difference is almost 19 points in **Acc@1** across datasets. Then we can conclude on two things; initializing LM Head from scratch improves the overall scores and makes our model more robust to domain shifts.

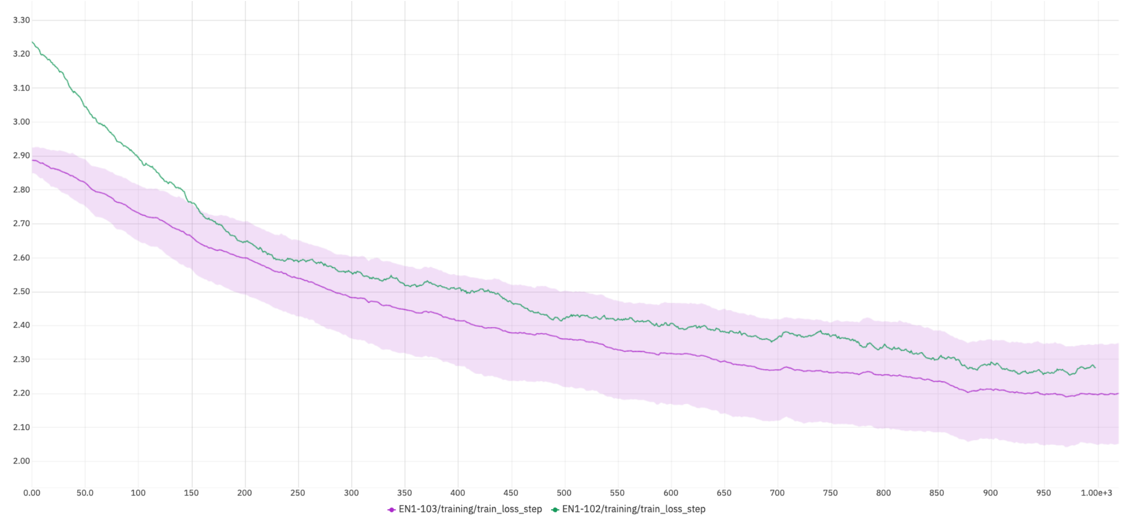
### 5.5.3 Initializing the Contextualized Entity Tokens

As a default, we add new tokens to our Tokenizer where each token corresponds to an entity. With this addition, we are automatically increasing the embedding layer size of our Transformer model. However, newly added embeddings are initialized randomly. Since we are using pre-trained Transformer models, text tokens are already learned and our model can make sense of them without fine-tuning. On the other hand, the new entity tokens are initialized from scratch. Thus, we experimented with the initializing method of entity tokens for making our model somehow aware of what they are about even at the beginning of the fine-tuning.

To achieve semantically meaningful entity tokens, we exploit external information. The 'description' of each entity was put through the initial model and the embedding for the $[CLS]$ token was extracted. The description is simply the first sentence of the Wikipedia page of the given entity. The $[CLS]$ token is widely used for sentence representation. Instead of random initializing of embeddings in the layer, we replaced them with the $[CLS]$ token embeddings of their descriptions.

In Figure 5.4, the training loss for our method and randomly initialized versions can be seen. Figure 5.4 shows that randomly initialized entity tokens are easier to predict since they are providing better loss. This might be due to the fact that randomly initialized embeddings are giving more flexibility to our model to learn better token representations for the entity tokens.

Figure 5.4 The loss curves for two training of EntityBERT model. Purple line (—) is showing the loss for an EntityBERT model which has randomly initialized entity tokens. The green line (—) is showing the loss for an EntityBERT model with entity tokens initialized with $[CLS]$ tokens of their description.




## 5.6 Results

The EntityBERT model was proposed by us and we did a few experiments with its components and hyperparameters which are discussed in detail earlier. These hyperparameter search trainings were between 10-15 hours of quick runs to see the effect of the tested variable (a component or hyperparameter). After deciding on the best practice in the experimental setting of EntityBERT training, we trained a final model for a longer period. We trained our model with a batch size of 128 and used a gradient accumulation step of 2 which makes the effective batch size 256. The learning rate was set to 1e-5. The masking probabilities for `ent_id_prob`, `ent_mask_prob` and `text_mask_prob` were set to 0.8, 0.4, and 0.3, respectively. This training took around 60 hours on a single GPU of 24 GB.
The BLINK and LUKE models were trained by following the instructions the authors provide.

These Neural Entity Linking models we explained in the earlier chapters were trained and evaluated in the same datasets. Tables 5.14, 5.14 and 5.16 shows the results for different *Acc@K* metrics with each model for Turkish Wikipedia test set, Mewsli-9 Turkish set, and TW-SUNLP-EL set, respectively. Table 5.17 shows the overall results of these models on the three different datasets.

Table 5.14 BLINK, LUKE, EntityBERT results on Turkish Wikipedia test set.

| Model | Acc@1 | Acc@4 | Acc@8 | Acc@16 | Acc@32 | Acc@64 |
|---|---|---|---|---|---|---|
| $\text{EntityBERT}_{\text{local}}$ | 51.32 | 69.74 | 77.29 | 85.55 | 90.24 | 94.69 |
| $\text{EntityBERT}_{\text{global}}$ | 50.59 | 68.30 | 76.09 | 83.49 | 89.80 | 94.19 |
| $\text{LUKE}_{\text{local}}$ | 91.00 | 99.76 | 99.86 | 99.86 | 99.90 | 99.95 |
| $\text{LUKE}_{\text{global}}$ | 91.46 | 97.90 | 98.18 | 98.42 | 98.84 | 99.08 |
| $\text{BLINK}_{\text{bi-encoder}}$ | 95.86 | 99.62 | 99.77 | 99.84 | 99.89 | 99.96 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 96.22 | 99.66 | 99.77 | 99.90 | 99.93 | 99.96 |

Table 5.15 BLINK, LUKE, EntityBERT results on Mewsli-9 Turkish set.

| Model | Acc@1 | Acc@4 | Acc@8 | Acc@16 | Acc@32 | Acc@64 |
|---|---|---|---|---|---|---|
| $\text{EntityBERT}_{\text{local}}$ | 49.59 | 52.89 | 62.90 | 72.29 | 81.09 | 88.30 |
| $\text{EntityBERT}_{\text{global}}$ | 47.50 | 51.76 | 61.93 | 71.30 | 82.10 | 88.23 |
| $\text{LUKE}_{\text{local}}$ | 91.45 | 97.90 | 98.18 | 98.42 | 98.84 | 99.08 |
| $\text{LUKE}_{\text{global}}$ | 91.14 | 99.76 | 99.86 | 99.86 | 99.90 | 99.95 |
| $\text{BLINK}_{\text{bi-encoder}}$ | 96.38 | 99.96 | 99.96 | 100.00 | 100.00 | 100.00 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 97.12 | 99.96 | 99.96 | 100.00 | 100.00 | 100.00 |

Table 5.16 BLINK, LUKE, EntityBERT results on TW-SUNLP-EL set.

| Model | Acc@1 | Acc@4 | Acc@8 | Acc@16 | Acc@32 | Acc@64 |
|---|---|---|---|---|---|---|
| $\text{EntityBERT}_{\text{local}}$ | 32.19 | 46.70 | 57.89 | 68.40 | 79.60 | 86.79 |
| $\text{EntityBERT}_{\text{global}}$ | 31.60 | 49.30 | 59.89 | 69.70 | 78.29 | 88.80 |
| $\text{LUKE}_{\text{local}}$ | 81.48 | 98.14 | 98.14 | 98.14 | 98.14 | 98.14 |
| $\text{LUKE}_{\text{global}}$ | 81.48 | 98.14 | 98.14 | 98.14 | 98.14 | 98.14 |
| $\text{BLINK}_{\text{bi-encoder}}$ | 85.53 | 98.03 | 98.03 | 98.03 | 99.34 | 99.70 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 87.26 | 98.67 | 98.67 | 98.93 | 99.34 | 99.70 |

Table 5.17 Performances of BLINK, LUKE, and EntityBERT on Turkish Wikipedia test set (**Test**), *Mewsli-9 Turkish* set (**M9**), and TW-SUNLP-EL (**TW**) set. We are using the Accuracy@1 score for this table.

| Model | Test | M9 | TW |
|---|---|---|---|
| $\text{BLINK}_{\text{bi-encoder}}$ | 95.86 | 96.38 | 85.53 |
| $\text{BLINK}_{\text{cross-encoder}}$ | 96.22 | 97.12 | 87.26 |
| $\text{LUKE}_{\text{local}}$ | 91.00 | 91.45 | 81.48 |
| $\text{LUKE}_{\text{global}}$ | 91.46 | 91.14 | 81.48 |
| $\text{EntityBERT}_{\text{local}}$ | 51.32 | 49.59 | 32.19 |
| $\text{EntityBERT}_{\text{global}}$ | 50.59 | 47.50 | 31.60 |

All these results will be used to answer our initial research questions in the next chapter.

# 6. CONCLUSION

In this thesis, we explored the Neural Entity Linking task by exploiting pretrained transformer-based models in Turkish. We adapt two advanced Neural Entity Linking systems originally designed for English, to Turkish. Additionally, we deliver a new simple Neural Entity Linking system called EntityBERT as a reference point for comparison.

The first model we adapted is called BLINK. The BLINK model is a straightforward, two-stage solution for the Entity Linking problem. It employs separate models, both based on fine-tuned BERT architectures, to handle candidate generation and re-ranking. In the first stage, a bi-encoder is used to retrieve candidates in a dense space, where the mentioned context and entity descriptions are independently embedded. Subsequently, in the second stage, the top N candidates are re-ranked using a cross-encoder model. Overall, BLINK offers a conceptually simple yet effective approach for Entity Linking tasks.

BLINK model exploits the entity mention text to model the context and mention, and external knowledge (Wikipedia titles and content of entities) to model entities. Overall, it uses three different transformer models; two for the bi-encoder and one for the cross-encoder.

Our experiment results in Table 5.1, 5.2 and 5.3, show that the BLINK model is extremely powerful for a relatively small set of entities in Turkish.

The second model we adapted is called LUKE. The LUKE model presents a novel approach to obtaining contextualized representations of words and entities using a bidirectional transformer. The model treats words and entities as separate tokens and generates contextualized representations for both. To train the model, a new pretraining task based on BERT's masked language model is used. This task involves predicting masked words and entities in a large entity-annotated corpus sourced from Wikipedia.

Moreover, the paper introduces an entity-aware self-attention mechanism that extends the transformer's self-attention mechanism by considering the token types (words or entities) when computing attention scores.

Different than the BLINK model, LUKE does not use any external data. It only uses

the entity mention context for modeling the entity representations. Moreover, while BLINK uses three different pretrained models for its aggregative pipeline, LUKE is a single model that handles the Entity Linking problem without the need for a complex inference pipeline or reranking module. It can be seen in Table 5.6, 5.7 and 5.8 that LUKE can achieve powerful results for Turkish as well.
As it is mentioned above, LUKE is a lighter model in terms of model size and complexity of inference. We implemented an even simpler model called EntityBERT. This model exploits the Masked Language Modelling task by just manipulating the input to the prertrained transformer model. It is learning entity representations from the context they appear like it does for words. It does not even use the entity mention text. In Section 5.4, we discussed a few experiments and their results. Overall, the EntityBERT model cannot perform as well as the previously mentioned models.
We conducted all these experiments to answer our research questions. Below, you can see these questions and the answers we provide.

> **RQ1:** Which type of Neural Entity Linking model, either dense-retrieval centric or contextualized-learning centric, would demonstrate superior performance on a small-scale Turkish dataset?
>
> **Answer:** To answer this question, we trained and evaluated BLINK and LUKE models on the same sets of Turkish datasets. While BLINK is the state-of-the-art model for transformer-based dense-retrieval approaches, LUKE is the state-of-the-art model that exploits only the contextualized-learning power of transformer models. Our results in Table 5.17 show that the BLINK model outperforms the LUKE model in the context of small-scale Turkish data.
>
> **RQ1.1:** How would these models be influenced by domain and genre shifts when evaluated on different datasets?
>
> **Answer:** To answer this question, we evaluated our models on three different datasets. The first dataset is the Turkish Wikipedia test set that we excluded from the Turkish Wikipedia dump that was used for training. Naturally, this set is sharing the same genre and domain as the training set. Therefore, the scores obtained in this set can be seen as the reference point. The second set is called Mewsli-9 Turkish. It is a collection of documents from Wikinews. It differs from the training set by its genre. While the training set consists of Wikipedia articles, Mewsli-9 consists of news articles. Finally, the third set is TW-SUNLP-EL set. We enriched a subset of the TW-SUNLP dataset, which is a collection of Turkish tweets for the Turkish NER task, with entity links. This set differs from the training set in terms of domain and writing style

(informal/formal).
Our results show that in both models the genre change does not affect the model performances. When the cross-encoder is used BLINK model even performs better in Mewsli-9 Turkish set compared to the Turkish Wikipedia test set. However, the domain shift performs both models similarly by decreasing the scores by around 10%. Additionally, the decrease is more than 20% for the EntityBERT model.

**RQ1.2:** Is it feasible to leverage the contextualized learning capabilities offered by transformer models, along with the Masked Language Modeling task, to tackle Neural Entity Linking in the Turkish language?

**Answer:** To answer this question we implemented a new model called EntityBERT which is inspired by LUKE. LUKE model uses a new pertaining task based on Masked Language Modelling. It tries to predict masked tokens and entities in the given sequence. We developed the EntityBERT model to see if it is feasible to directly use the Masked Language Modelling task and contextualized learning capabilities of transformer models for learning entity representations and solving Entity Linking. Our results show that this model alone does not perform even near the previous models we discussed. It is almost 40-45% behind the state-of-the-art models in the Turkish Wikipedia test set and this difference increases when the genre and domain shifts occur.

## 6.1 Future Work

Our results show that in the small-scale Turkish data, the dense-retrieval-based model BLINK performs better than the pure transformer-based model LUKE. However, as it was mentioned, the BLINK model uses a complex inference pipeline compared to LUKE and external knowledge. Moreover, the original results from their papers show that the LUKE model outperforms BLINK on full Wikipedia training. One of the future steps we can take would be scaling these models to the full Turkish Wikipedia and comparing their performances in terms of metrics and efficiency.
On the other hand, to improve the performance of the EntityBERT model we can work on the architecture to enable it to be exposed to the entity mention text with the context. Additionally, since it is a straightforward model with a simple implementation, we can try to use EntityBERT as a neural candidate generation

component which will be followed by a simple cross-encoder module where we exploit external data as well. With this method, we can 1) expose our model to entity mention and external knowledge similar to BLINK and 2) we can exploit the transformer architecture as a simple but efficient candidate generation tool similar to LUKE. This can bring a model which is a mix of these two powerful models.
To address the performance decrease with domain shift, we can do these experiments with pretrained transformer models which have Twitter data in their training corpus. Additionally, we can extend the TW-SUNLP-EL set for a bigger coverage and use a subset of it for fine-tuning the models to see if it will help with the performance decrease.

# BIBLIOGRAPHY


Barba, E., Procopio, L., & Navigli, R. (2022). ExtEnD: Extractive entity disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 2478–2488)., Dublin, Ireland. Association for Computational Linguistics.

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer.

Botha, J. A., Shan, Z., & Gillick, D. (2020). Entity linking in 100 languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 7833–7845).

Broscheit, S. (2019). Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, (pp. 677–685)., Hong Kong, China. Association for Computational Linguistics.

Bunescu, R. & Paşca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 9–16)., Trento, Italy. Association for Computational Linguistics.

Cao, N. D., Izacard, G., Riedel, S., & Petroni, F. (2021). Autoregressive entity retrieval.

Çarık, B. & Yeniterzi, R. (2022). A Twitter corpus for named entity recognition in Turkish. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, (pp. 4546–4551)., Marseille, France. European Language Resources Association.

Cucerzan, S. (2007a). Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, (pp. 708–716)., Prague, Czech Republic. Association for Computational Linguistics.

Cucerzan, S. (2007b). Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, (pp. 708–716)., Prague, Czech Republic. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Févry, T., FitzGerald, N., Soares, L. B., & Kwiatkowski, T. (2020). Empirical evaluation of pretraining strategies for supervised entity linking.

Ganea, O.-E. & Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 2619–2629)., Copenhagen, Denmark. Association for Computational Linguistics.

Gillick, D., Kulkarni, S., Lansing, L., Presta, A., Baldridge, J., Ie, E., & Garcia-Olano, D. (2019). Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning*

*(CoNLL)*, (pp. 528–537)., Hong Kong, China. Association for Computational Linguistics.

Guo, S., Chang, M.-W., & Kiciman, E. (2013). To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 1020–1030)., Atlanta, Georgia. Association for Computational Linguistics.

Harandizadeh, B. & Singh, S. (2020). Tweeki: Linking named entities on Twitter to a knowledge graph. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, (pp. 222–231)., Online. Association for Computational Linguistics.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., & Wang, H. (2013). Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (pp. 30–34)., Sofia, Bulgaria. Association for Computational Linguistics.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., & Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, (pp. 782–792)., Edinburgh, Scotland, UK. Association for Computational Linguistics.

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, *7*(3), 535–547.

Kalender, M. & Korkmaz, E. E. (2018). Thinker - entity linking system for turkish language. *IEEE Transactions on Knowledge and Data Engineering*, *30*(2), 367–380.

Kar, R., Reddy, S., Bhattacharya, S., Dasgupta, A., & Chakrabarti, S. (2018). Task-specific representation learning for web-scale entity disambiguation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Klie, J.-C., Bugert, M., Boullosa, B., Eckart de Castilho, R., & Gurevych, I. (2018). The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, (pp. 5–9)., Santa Fe, New Mexico.

Kolitsas, N., Ganea, O.-E., & Hofmann, T. (2018). End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, (pp. 519–529)., Brussels, Belgium. Association for Computational Linguistics.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Ling, J., FitzGerald, N., Shan, Z., Soares, L. B., Févry, T., Weiss, D., & Kwiatkowski, T. (2020). Learning cross-context entity representations from text.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Logeswaran, L., Chang, M.-W., Lee, K., Toutanova, K., Devlin, J., & Lee, H. (2019).

Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (pp. 3449–3460)., Florence, Italy. Association for Computational Linguistics.

Loshchilov, I. & Hutter, F. (2019). Decoupled weight decay regularization.

Möller, C., Lehmann, J., & Usbeck, R. (2021). Survey on english entity linking on wikidata.

Ratinov, L., Roth, D., Downey, D., & Anderson, M. (2011a). Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (pp. 1375–1384)., Portland, Oregon, USA. Association for Computational Linguistics.

Ratinov, L., Roth, D., Downey, D., & Anderson, M. (2011b). Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, (pp. 1375–1384)., USA. Association for Computational Linguistics.

Rizzo, G., van Erp, M., Plu, J., & Troncy, R. (2015). Making sense of microposts (#microposts2016) named entity recognition and linking (neel) challenge. In *Workshop on Making Sense of Microposts*.

Schweter, S. (2020). Berturk - bert models for turkish.

Sun, T., Shao, Y., Qiu, X., Guo, Q., Hu, Y., Huang, X., & Zhang, Z. (2020). CoLAKE: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, (pp. 3660–3670)., Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sun, Y., Lin, L., Tang, D., Yang, N., Ji, Z., & Wang, X. (2015). Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, (pp. 1333–1339). AAAI Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wu, L., Petroni, F., Josifoski, M., Riedel, S., & Zettlemoyer, L. (2020). Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 6397–6407)., Online. Association for Computational Linguistics.

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models. *Transactions of the Association for Computational Linguistics*, *10*, 291–306.

Yamada, I., Asai, A., Shindo, H., Takeda, H., & Matsumoto, Y. (2020). LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 6442–6454)., Online. Association for Computational Linguistics.

Yamada, I., Shindo, H., Takeda, H., & Takefuji, Y. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, (pp. 250–259)., Berlin, Germany. Association for Computational Linguistics.

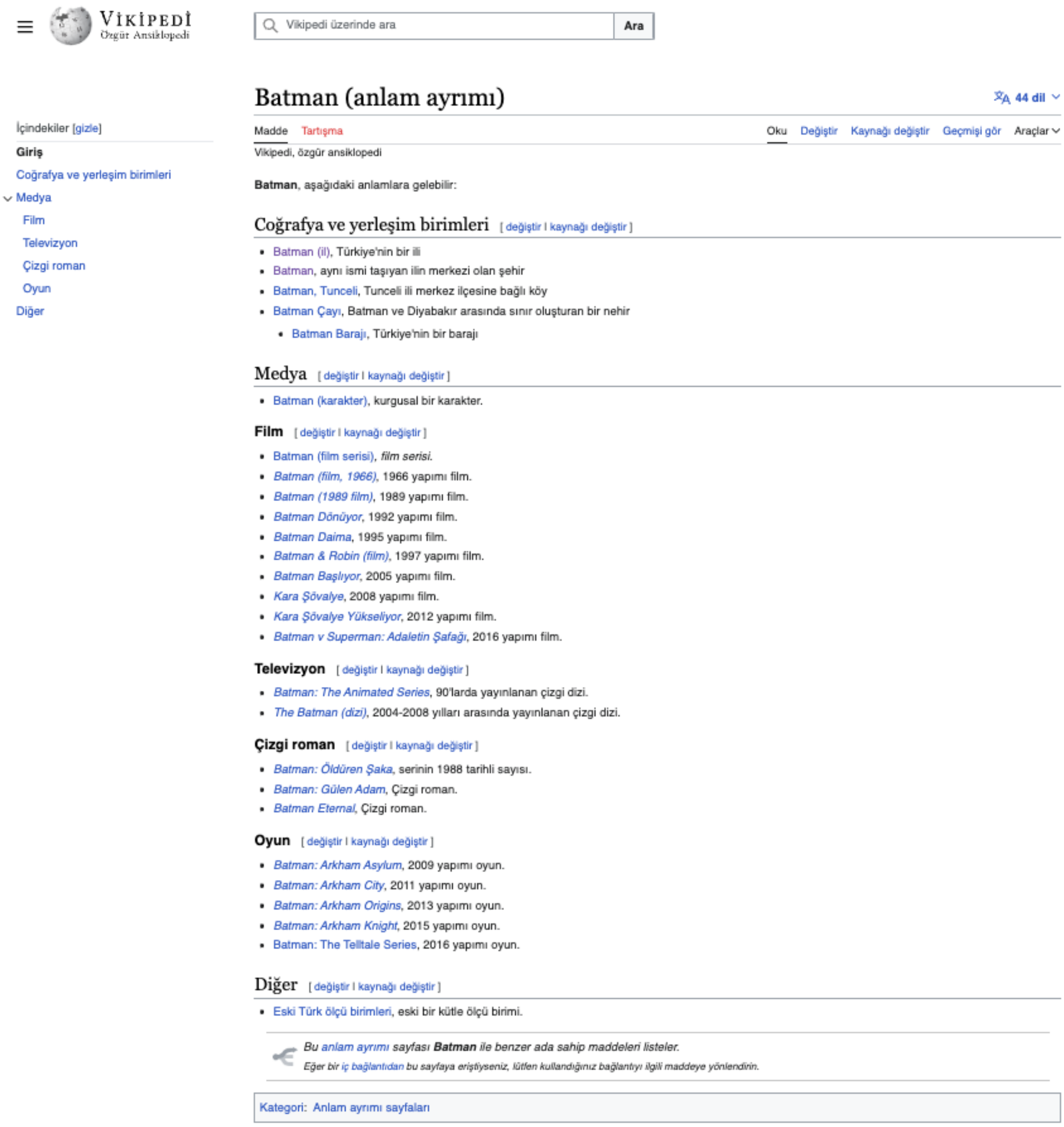
Yamada, I., Washio, K., Shindo, H., & Matsumoto, Y. (2022a). Global entity disambiguation with BERT. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 3264–3271)., Seattle, United States. Association for Computational Linguistics.

Yamada, I., Washio, K., Shindo, H., & Matsumoto, Y. (2022b). Global entity disambiguation with bert.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., & Liu, Q. (2019). Ernie: Enhanced language representation with informative entities.

# APPENDIX A

Figure A.1 A sample page for Turkish Wikipedia disambiguation pages. This page is for the term *Batman* which has a page for a city in Turkey, the famous movie series, the comic character, and even more.


ViKiPEDi
Özgür Ansiklopedi

Vikipedi üzerinde ara | Ara

İçindekiler [gizle]
Giriş
Coğrafya ve yerleşim birimleri
Medya
Film
Televizyon
Çizgi roman
Oyun
Diğer

Batman (anlam ayrımı)

44 dil

Madde Tartışma Oku Değiştir Kaynağı değiştir Geçmişi gör Araçlar

Vikipedi, özgür ansiklopedi

**Batman**, aşağıdaki anlamlara gelebilir:

Coğrafya ve yerleşim birimleri [ değiştir | kaynağı değiştir ]

- Batman (il), Türkiye'nin bir ili
- Batman, aynı ismi taşıyan ilin merkezi olan şehir
- Batman, Tunceli, Tunceli ili merkez ilçesine bağlı köy
- Batman Çayı, Batman ve Diyabakır arasında sınır oluşturan bir nehir
  - Batman Barajı, Türkiye'nin bir barajı

Medya [ değiştir | kaynağı değiştir ]

- Batman (karakter), kurgusal bir karakter.

**Film** [ değiştir | kaynağı değiştir ]

- Batman (film serisi), *film serisi.*
- *Batman (film, 1966)*, 1966 yapımı film.
- *Batman (1989 film)*, 1989 yapımı film.
- *Batman Dönüyor*, 1992 yapımı film.
- *Batman Daima*, 1995 yapımı film.
- *Batman & Robin (film)*, 1997 yapımı film.
- *Batman Başlıyor*, 2005 yapımı film.
- *Kara Şövalye*, 2008 yapımı film.
- *Kara Şövalye Yükseliyor*, 2012 yapımı film.
- *Batman v Superman: Adaletin Şafağı*, 2016 yapımı film.

**Televizyon** [ değiştir | kaynağı değiştir ]

- *Batman: The Animated Series*, 90'larda yayınlanan çizgi dizi.
- *The Batman (dizi)*, 2004-2008 yılları arasında yayınlanan çizgi dizi.

**Çizgi roman** [ değiştir | kaynağı değiştir ]

- *Batman: Öldüren Şaka*, serinin 1988 tarihli sayısı.
- *Batman: Gülen Adam*, Çizgi roman.
- *Batman Eternal*, Çizgi roman.

**Oyun** [ değiştir | kaynağı değiştir ]

- *Batman: Arkham Asylum*, 2009 yapımı oyun.
- *Batman: Arkham City*, 2011 yapımı oyun.
- *Batman: Arkham Origins*, 2013 yapımı oyun.
- *Batman: Arkham Knight*, 2015 yapımı oyun.
- Batman: The Telltale Series, 2016 yapımı oyun.

Diğer [ değiştir | kaynağı değiştir ]

- Eski Türk ölçü birimleri, eski bir kütle ölçü birimi.

*Bu anlam ayrımı sayfası **Batman** ile benzer ada sahip maddeleri listeler.*
*Eğer bir iç bağlantıdan bu sayfaya eriştiyseniz, lütfen kullandığınız bağlantıyı ilgili maddeye yönlendirin.*

Kategori: Anlam ayrımı sayfaları