

Zshot: An Open-source Framework for Zero-Shot Named Entity Recognition and Relation Extraction

Gabriele Picco
IBM Research Europe
gabriele.picco@ibm.com

Marcos Martinez Galindo
IBM Research Europe
marcos.martinez.galindo@ibm.com

Alberto Purpura
IBM Research Europe
alp@ibm.com

Leopold Fuchs
IBM Research Europe
leopold.fuchs@ibm.com

Vanessa Lopez
IBM Research Europe
vanlopez@ie.ibm.com

Hoang Thanh Lam
IBM Research Europe
t.l.hoang@ie.ibm.com

Abstract

The Zero-Shot Learning (ZSL) task pertains to the identification of entities or relations in texts that were not seen during training. ZSL has emerged as a critical research area due to the scarcity of labeled data in specific domains, and its applications have grown significantly in recent years. With the advent of large pretrained language models, several novel methods have been proposed, resulting in substantial improvements in ZSL performance. There is a growing demand, both in the research community and industry, for a comprehensive ZSL framework that facilitates the development and accessibility of the latest methods and pretrained models. In this study, we propose a novel ZSL framework called Zshot that aims to address the aforementioned challenges. Our primary objective is to provide a platform that allows researchers to compare different state-of-the-art ZSL methods with standard benchmark datasets. Additionally, we have designed our framework to support the industry with readily available APIs for production under the standard SpaCy NLP pipeline. Our API is extendible and evaluable, moreover, we include numerous enhancements such as boosting the accuracy with pipeline ensembling and visualization utilities available as a SpaCy extension. <https://youtu.be/Mhc1zJXKEJQ>

1 Introduction

Zero-Shot Learning (ZSL) is a machine learning field focused on the study of models able to classify objects or perform tasks that they have not experienced during training. This is achieved by leveraging additional information about the output classes, such as their attributes or descriptions.

ZSL has a wide range of potential applications, since it allows a model to generalize and adapt to new situations without requiring retraining or large amounts of labeled data. This can be particularly useful in real world applications where new classes or categories may be constantly emerging and it

would be infeasible to retrain the model every time. ZSL can also be used to classify and predict rare or minority classes that may not have a significant amount of labeled data available for training.

If we consider Named Entity Recognition (NER) – including classification and linking (NEL) – and Relation Extraction (RE) problems, recent ZSL methods [Aly et al. \(2021\)](#); [Wu et al. \(2020\)](#); [Chen and Li \(2021\)](#) leverage textual descriptions of entities or relations as additional information to perform their tasks. This additional input allows models to recognize previously unseen entities (or relations) in text, based on the provided descriptions. [Wu et al. \(2020\)](#) and [Aly et al. \(2021\)](#) provide examples of the effectiveness of descriptions in the zero-shot NER task. The same mechanism can also be applied to the RE task ([Chen and Li, 2021](#)) by providing descriptions of the relation between entity pairs. Figure 1 shows the input and output of a zero-shot NER and classification model such as SMXM ([Aly et al., 2021](#)). Leveraging entity descriptions, the model is able to disambiguate between mentions of the term "apple" – which could indicate a mention of the homonymous company, or the fruit – despite having been trained only on OntoNotes ([Weischedel et al., 2017](#)) classes. By projecting each token in the latent space of a pretrained Language Model (LM), the semantic distance between the label/description of a class and each token in a sentence can be used as a method assign tokens to a certain entity.

There are several variations to ZSL approaches, both for NER and RE. For example, [De Cao et al. \(2021\)](#) show how it is possible to frame the zero-shot NER task as an end-to-end autoregressive generation problem that exploits the probability distribution learned by the LM, while [Wu et al. \(2020\)](#) uses a combination of a bi-encoder and a cross-encoder for entity linking. In addition to the different configurations of NER and RE models – i.e. model architectures, training strategies and

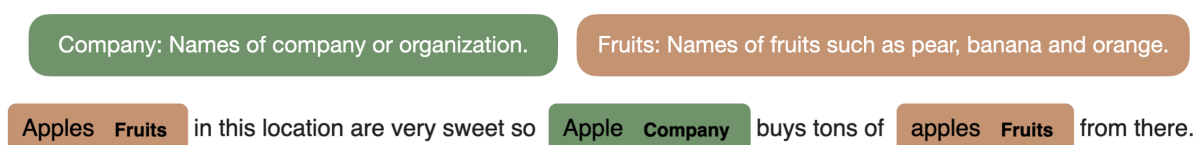


Figure 1: An example of a zero-shot NER prediction. Using textual descriptions, the model is able to differentiate between the two unseen classes.

datasets – these components often depend on other NLP tools to process texts and on each other, e.g. RE models often are not end-to-end and require entity spans in a text as part of their input.

2 Motivations and Contributions

The hidden complexity in running NER or RE experiments in the ZSL context and the lack of a shared framework and API to describe the inputs and outputs of such models hampers the reproducibility of many approaches, as well as the ability of the research community to evaluate new models against strong baselines under the same experimental conditions. More specifically, we identified the following major shortcomings for what concerns the ability to evaluate existing NER and RE approaches under the same experimental conditions.

Different assumptions about the task. Some NER and Entity Linking methods assume that the mentions to link to are given, while others are end-to-end. Similar considerations apply also to RE, where some approaches tackle the task as an end-to-end problem and others as a classification one. Furthermore, some approaches may be limited in terms of performance or scalability and might therefore provide an evaluation only on a restricted subset of a dataset or on a simplified version of their task. For example, entity linking models are often unable to link entities to a dataset the size of Wikipedia or Wikidata in the same way as many RE approaches cannot scale to hundreds of relation classes. These differences in the input-output configuration of NER and RE models impact the ability of researchers to evaluate new approaches against existing ones, either for the lack of clarity when describing their evaluation strategy or for the investment required to make the appropriate changes to the models configurations, in order to make them comparable under the same experimental conditions.

Lack of standard datasets. As the performance of ZSL approaches increases, better and more specific datasets are developed to help researchers evaluate increasingly challenging problems. The release of the FewRel and FewRel 2.0 datasets is an example of this refinement process (Gao et al., 2019). In the first version of FewRel, a RE model was expected to assign to each entity pair in a sentence a relation from a list. In other words, there was always a correct class for the model to pick to describe the relation between an entity pair. However, the most frequently occurring scenario in the real world is when two entities are not related or when the correct relation between the entities is missing among the options given by the model. This prompted the release of an updated version of the dataset, which includes the *no relation* class as an option for RE models to choose from. Similar considerations can be made for the evaluation of NER models and domain-specific ones. Another aspect which is often overlooked in many research papers is the lack of a shared implementation of a training, validation and test split for a dataset. This is especially important in the ZSL setting where entities or relation classes should not overlap between training/validation and test dataset splits. Often, zero-shot datasets are obtained from a random split of existing datasets, which were not originally designed to be used for the evaluation of zero-shot approaches and contain overlapping output classes in the respective training/validation and test set. In the process of transforming these datasets the exact split of classes is often not reported and this hampers the reproducibility of the evaluation results presented in a research work.

Different evaluation metrics. To compare a group of models under the same experimental conditions, different researchers will have to agree on a set of evaluation metrics to employ. This often happens when a new NLP task is introduced, together with a reference dataset and baselines. For NER and RE, the reference metric is often the Macro

F1 Score, but there might be models which have a better precision/recall balance depending on the task or that have been evaluated on different, more domain-specific metrics when first described to the research community. These differences make it harder for researchers to benchmark models across different datasets or domains and to evaluate the actual improvements of newly proposed approaches.

To tackle the above-described problems concerning the evaluation and benchmarking of zero-shot NER and RE models, we present Zshot. Zshot is an open-source, easy-to-use, and extensible framework for ZSL in NLP.

The main contributions of the Zshot framework are the following:

- Standardization and modularization: Zshot standardizes and modularizes the NER and RE tasks providing an easy to use and customize API for these models.
- Unification of NER and RE: these tasks are often considered separately in existing literature, however this is not the case in real world scenarios where the models are strongly interconnected. In Zshot, users can define unified pipelines for both tasks.
- Compatible with SpaCy (Honnibal and Montani, 2017) and the HuggingFace library (Wolf et al., 2019): users define pipelines following SpaCy style NLP pipelines and models are hosted by HuggingFace.
- Evaluation: provides an easy to use and extend evaluation framework for different models and pipelines on standard benchmark datasets for NER and RE.
- Visualization: Zshot builds on displaCy capabilities as a visualization tool to display results of NER annotations and RE models.
- Ensemble pipeline: Zshot provides simple API for ensembling of NER or RE pipelines using different entity or relation descriptions or models, yielding more accurate results than standalone systems.
- Open source: the open source community can customise and extend Zshot adding new models, evaluation metrics and datasets.

```

1 nlp_config = PipelineConfig(
2     entities=[
3         Entity(
4             name="Company",
5             description="Names of
6                 company or
7                 organisation"
8         ),
9         Entity(
10            name="Fruits",
11            description="Names of
12                fruits such as pear,
13                banana and orange"
14        )
15    ],
16    linker=LinkerSMXM(),
17 )

```

Listing 1: Python example of a Zshot pipeline configuration.

The Zshot library implements a pipeline defined by 3 components i.e., *Mention Detection*, *Entity Linking* and *Relation Extraction* and is compatible with SpaCy (Honnibal and Montani, 2017). It extends the popular displaCy tool for zero-shot entity and relation visualization, and defines an evaluation pipeline that validates the performance of all components, making it compatible with the popular HuggingFace library (Wolf et al., 2019). Zshot aims to provide an easy-to-use solution that can be integrated in production pipelines, while at the same time providing researchers with a framework to easily contribute, validate and compare new state-of-the-art approaches. We open source all the code, models, metrics and datasets used.¹

3 Design and Implementation

Figure 2 shows a high-level overview of the Zshot pipeline composed by three main modules i.e., *Mention Detection*, *Entity Linking* and *Relation Extraction*. The pipeline accepts a configuration object that allows to select the output classes of mentions, entities and relationships of interest and the models to use to perform each task (see Listing 1). Then, each of the library modules adds annotations that can be used at a later stage, extending a SpaCy NLP pipeline. Components like the Entity Linker and the Relation Extractor can also be end-to-end, in which case the pipeline automatically skips the unnecessary previous steps. Zshot also automatically manages batching, parallelization and the device on which to perform the computation. All options are configurable and modules are customizable.

¹Zshot: <https://github.com/IBM/zshot>

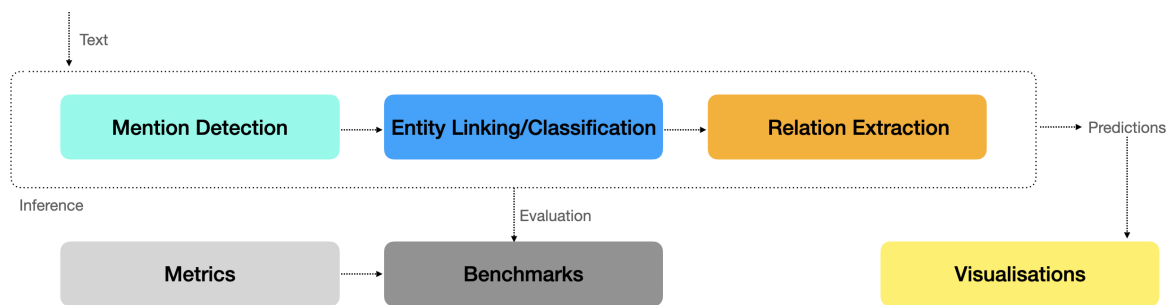


Figure 2: High-level architecture of Zshot. The Entity Linker and the Relation Extractor can use the predictions of previous components. For end-to-end models, the pipeline will skip the previous steps.

Listing 1 contains a sample pipeline configuration for NER, the same used to produce the example shown in Figure 1. The configuration defines the two entity classes to be identified, providing a title and a textual description for each of them. It is also specified to use the SMXM end-to-end linker (see Section 3.2 for more details).

In the remainder of this section, we will dive deeper into the details of each of the building blocks of Zshot.

3.1 Mention Detection

Mention detection is the task that consists in identifying spans of text that may contain entities in a defined set of classes of interest. In our framework, mention detection is used as a pre-processing step for entity linkers (see 3.2) that require mentions as input, such as (Wu et al., 2020) and (Cao, 2021).

Zshot currently supports six mentions extractors. Four reuse existing generic annotators (i.e., Flair (Akbik et al., 2019) and SpaCy) while the other two are sensitive to the defined mentions/entities classes (i.e., TARS (Halder et al., 2020) and SMXMs).

3.2 Entity Linking and Classification

Entity classification is the process of identifying the type of a given mention in the text. For example, if a text mentions "Barack Obama", entity classification would determine that this refers to a person, but would not specify which person. Entity linking, also known as named entity disambiguation, is the process of identifying and disambiguating mentions of entities in a text, linking them to their corresponding entries in a knowledge base or a dictionary. For example, given "Barack Obama", entity linking would determine that this refers to the specific person with that name (one of the presidents of the United States) and not any other person

or concept with the same name. In Zshot we introduce the Linkers. Linkers can perform both entity classification and entity linking, depending on the entities being used. For simplicity, we use entity linking as the name of the task, but this comprises both entity classification and entity linking. Entity linking can be useful for a variety of natural language processing tasks, such as information extraction, question answering, and text summarization. It helps to provide context and background information about the entities mentioned in the text, which can facilitate a deeper understanding of the content. There are several techniques that can be used for entity linking, including dictionary-based methods, rule-based methods, and machine learning-based methods.

Zshot currently supports four Entity linking and classification methods (Linkers): Blink (Wu et al., 2020), GENRE (De Cao et al., 2021), TARS and SMXM (Aly et al., 2021).

3.3 Wikification

Two of the entity linkers, Blink and GENRE, can scale to very large knowledge bases and in Zshot they can be used to perform *Wikification* out-of-the-box. Wikification is the process of adding *wikilinks* to a piece of text, which allows readers to easily navigate to related articles or pages within a wiki or other online encyclopedia. A wikilink is a hyperlink that is used within Wikipedia – or another online encyclopedia – to link to other pages within the same resource. Wikification is often used to provide context and background information about the concepts and entities mentioned in a piece of text. It can also be used to create a network of interconnected articles within Wikipedia or another online encyclopedia, which makes it easier for readers to explore related topics and gain a deeper understand-

ing of the content.

3.4 Relation Extraction

Relation extraction (or classification) is the task inferring the relation between an entity pair within a portion of text. This task is often framed as a classification problem where a text with entity mentions are given in input to a classifier which is trained to recognize the correct relation type connecting the entities. In the literature, we can also find variants of this approach such as (Ni et al., 2022) which operate in an end-to-end fashion, without requiring entity mentions. ZSL approaches for RE often receive in input a set of candidate of relation descriptions and match them with each entity pair in a text.

Currently, Zshot supports one relation classification model i.e., ZS-BERT (Chen and Li, 2021). This model performs relation classification by first embedding relation descriptions using a sentence embedding model and then comparing it with an entity pair embedding computed with a LM fine-tuned for this task. Entity pairs are finally associated to the closest relation class in the embedding space in terms of cosine similarity.

3.5 Ensembling

There are different pretrained linkers and mention extractors. These models are pretrained with different data, using different training methods and neural architectures. We provide an easy way to combine the predictions from these models so that we can achieve a more robust result by leveraging the strengths of the diverse set of models.

Besides the ensembling of linkers, we also support an API for making an ensemble of pipelines with different entity/relation descriptions. We discovered that the accuracy of ZSL models is very sensitive to provided entity/relation descriptions. Combining prediction from pipelines with different descriptions potentially provides significant improvement. An example of the API to make an ensemble of linkers, and mention extractors with various descriptions is demonstrated in the listing 5 in the Appendix.

3.6 Customization

One of the most important parts of a framework is to allow the community to create and share new components, which results in improving the models' performance and the appearance of new ones. Zshot allows users to create new components easily,

by extending one of the abstract classes (depending on the type of component: mentions extractor, linker or relation extractor). The user just has to implement the predict method to create a new component, and Zshot will take care of the rest, including adding the result to the *SpaCy* document or adding it to the pipeline. Listing 3 shows a simple example of how to implement a custom mentions extractor that will extract as mentions all the words that contain the letter *s*. The predict method takes as input a list of *SpaCy* documents and return, for each document, a list of Zshot Span, with the boundaries of the mention in the text. We encourage the users to develop new components, models, and datasets and share them with the community.

4 Visualization

NER and RE models are broadly used in multiple and diverse NLP pipelines, from Knowledge Graph generation and population to Information Extraction systems. However, for development or research, NER and RE models are sometimes the final output of the process. For these reasons, an appealing visualization of the entities and relations extracted by a model is important. Visualization helps users to rapidly assess what entities and relations have been extracted, and allows them make changes to improve their models. In Zshot, we extend *displaCy*, a *SpaCy* tool for visualization. When using *displaCy* for NER with custom entities, the visualization shows the entities, but all of them have the same color, so it's not easy for the user to see the entities detected. We expanded the capabilities of the library to support distinct colors for different entities (see Figure 3).

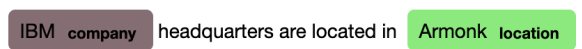


Figure 3: Visualization using Zshot version of *displaCy*.

At the time of writing, we could not find a visualization tool for RE in *displaCy*,² as RE is not supported in *SpaCy*. In Zshot, we extend *displaCy* to support the visualization of edges between entity pairs in a sentence as shown in Figure 4.

5 Evaluation

Evaluation is key to improve the performance of a system. To assess the performance of NER and RE models, Zshot provides an evaluation module. This

²<https://spacy.io/usage/visualizers>

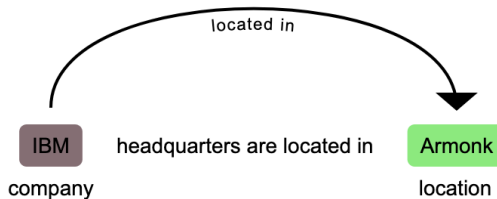


Figure 4: An example of RE visualization in Zshot.

module includes an interface to different datasets i.e., OntoNotesZS (Pradhan et al., 2013), Med-MentionsZS (Mohan and Li, 2019) for NER and FewRel (Han et al., 2018) for RE. All the datasets included have been preprocessed to assure that they can be used in the ZSL setting i.e., the entities/relations in the training or validation splits of these datasets are not overlapping with each other and are not included in the respective test sets to ensure a correct evaluation. These datasets are managed using the Huggingface datasets library,³ which makes it easier to add new datasets using the Huggingface Hub⁴. In Zshot, we use the evaluate package⁵, implementing evaluators for the Mentions Extraction, Entity Linking and RE tasks. On top of the evaluators, we added a function that receives a SpaCy NLP model with a Zshot configuration and the splits of the predefined datasets, and it evaluates the model on all datasets configurations. This function returns the evaluation results both as a table with the results as a string object or as a Python dictionary. As a default option, we use the SeqEval package (Nakayama, 2018) to compute the evaluation metrics, including Accuracy, Precision, Recall, F1-Score Macro and F1-Score Micro. However, using the evaluate package, it is also easy to define and use custom metrics extending the same package. Table 1 shows an example of the evaluation results format for the OntoNotes validation set, as it is returned by Zshot.

6 Conclusion and Future Work

In this paper, we described Zshot, a framework for zero-shot NER and RE in NLP. ZSL is a growing area of research in NLP. The increasing number of research works published every year and the difficulties associated to the lack of standardization

³<https://github.com/huggingface/datasets>

⁴<https://huggingface.co/datasets>

⁵<https://github.com/huggingface/evaluate>

Metric	ontonotes-test
overall_precision_macro	20.96%
overall_recall_macro	48.15%
overall_f1_macro	29.12%

Table 1: Example of result in Zshot for SMXM linker over the validation set of OntoNotesZS. These are only some metrics reported for one model, to see the whole result please check Table 2 in the Appendix, with a comparison between two different linkers.

motivated us to develop this framework.

Our work aims at standardizing experimental zero-shot pipelines to improve the reproducibility of existing approaches and facilitate the comparison with new ones. We defined a customizable interface based on the popular SpaCy library. This allows developers and researchers already familiar with this popular Python library to quickly try out zero-shot NLP approaches, while more experienced users can extend it to include new models. We also provide an evaluation package to aid the evaluation and comparison of NER and RE models on different datasets through standard evaluation metrics, which can be further customized and expanded by users. Finally, we extended the displaCy library to support visualizations of recognized entities and the relations between them as edges between entity spans. We open source all our code, models, metrics and datasets.

In the future, we plan to increase the number of supported models and datasets for NER and RE. We also aim to increase the efficiency of these models so that they could be employed by NLP practitioners in real world applications.

Limitations

Zshot is a SpaCy extension for ZSL. It currently supports models for Mention Detection, Entity Linking and Relation Extraction allowing users to evaluate the supported models and to visualize their outputs. The limitations of our framework are strongly dependent on these models trained on limited amounts of data (in English) for research purposes. Therefore, their results might not be reliable on certain domain-specific scenarios which were not included in the training data and may contain biases. Some models are also more scalable and efficient than others. The efficiency of a model will depend on its implementation. We focus on providing a standard API and an efficient framework based on SpaCy to run these models. The

pretrained models are required to fine-tuned with in-domain training classes, even non-overlapping with testing classes, generalization of these models in new domains is considered as future work.

Ethics Statement

Zshot is an opensource, easy-to-use, and extensible framework for ZSL. These models may contain bias and cause ethical concerns, as it can be seen in Figure 5 (see Appendix), where one of the models supported in Zshot assigns an entity label in a biased way, being based on the gender of the name. Bias in NLP models is a common issue ([Stanczak and Augenstein, 2021](#)), and a lot of efforts focus on mitigating this problem, ([Sun et al., 2019](#)). Zshot is a framework that aims at standardizing and facilitating the use zero-shot NLP models, and it does not increase nor decrease their bias. We encourage users to assess the bias of NLP models prior to employing them in any downstream task and to validate their results depending on the application scenario to eliminate any potentially negative impact.

References

- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.
- Rami Aly, Andreas Vlachos, and Ryan McDonald. 2021. Leveraging type descriptions for zero-shot named entity recognition and classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1516–1528, Online. Association for Computational Linguistics.
- Rui Cao. 2021. Holistic interpretation in locative alternation – evidence from self-paced reading. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 543–550, Shanghai, China. Association for Computational Linguistics.
- Chih-Yao Chen and Cheng-Te Li. 2021. ZS-BERT: towards zero-shot relation extraction with attribute representation learning. *CoRR*, abs/2104.04697.
- Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6250–6255, Hong Kong, China. Association for Computational Linguistics.
- Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-aware representation of sentences for generic text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3202–3213, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- Sunil Mohan and Donghui Li. 2019. Medmentions: A large biomedical corpus annotated with {umls} concepts. In *Automated Knowledge Base Construction (AKBC)*.
- Hiroki Nakayama. 2018. sequeval: A python framework for sequence labeling evaluation. Software available from <https://github.com/chakki-works/sequeval>.
- Jian Ni, Gaetano Rossiello, Alfio Gliozzo, and Radu Florian. 2022. A generative model for relation extraction and classification. *arXiv preprint arXiv:2202.13229*.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.
- Karolina Stanczak and Isabelle Augenstein. 2021. A survey on gender bias in natural language processing. *CoRR*, abs/2112.14168.
- Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.
- Ralph M. Weischedel, Eduard H. Hovy, Mitchell P. Marcus, and Martha Palmer. 2017. Ontonotes : A large training corpus for enhanced processing.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

A Appendix: Code Examples

We report below some examples of use of the Zshot framework:

- Listing 2 shows an example of how to use Zshot to compute both entities and relations and to visualize RE, as shown in Figure 4.
- Listing 3 shows an example of how to create a custom mentions extractor in Zshot.
- Figure 5 reports an example of possible biases contained in NLP models for relation classification.
- Listing 5 shows an example of how to use Zshot to make an ensembles of existing pretrained linkers and descriptions. In this specific example, two linkers are used together with two different descriptions of the entity *fruits*.
- Listing 4 shows an example of how to use Zshot to annotate a sentence with entities and relations. Figure 6 shows the result.

```
1 # Import SpaCy
2 import spacy
3
4 # Import PipelineConfig for Zshot configuration and displacy for visualization
5 from zshot import PipelineConfig, displacy
6 # Import Entity and Relation data models
7 from zshot.utils.data_models import Entity, Relation
8 # Import LinkerSMXM for end2end NER
9 from zshot.linker import LinkerSMXM
10 # Import ZSRC for RE
11 from zshot.relation_extractor import RelationsExtractorZSRC
12
13 # Create/Load an NLP model
14 nlp = spacy.load("en_core_web_sm")
15 # Create a Zshot configuration with SMXM, ZSRC and some entities and relations
16 nlp_config = PipelineConfig(
17     linker=LinkerSMXM(),
18     relations_extractor=RelationsExtractorZSRC(thr=0.1),
19     entities=[
20         Entity(name="company", description="The name of a company"),
21         Entity(name="location", description="A physical location"),
22     ],
23     relations=[
24         Relation(name='located in', description="If something like a person, a
25         building, or a company is located in a particular place, like a city, country
26         of any other physical location, it is present or has been built there")
27     ]
28 )
29 # Add Zshot to SpaCy pipeline
30 nlp.add_pipe("zshot", config=nlp_config, last=True)
31
32 # Run the pipeline over a text
33 text = "IBM headquarters are located in Armonk."
34 doc = nlp(text)
35 # Visualize the result
36 displacy.render(doc, style='rel')
```

Listing 2: Python example of using Zshot displacy.

```

1 # Import SpaCy
2 import spacy
3 # Import types needed for typing
4 from typing import Iterable
5 from spacy.tokens import Doc
6 # Import Zshot PipelineConfig
7 from zshot import PipelineConfig
8 # Import Zshot Span to save the results
9 from zshot.utils.data_models import Span
10 # Import abstract class
11 from zshot.mentions_extractor import MentionsExtractor
12
13 # Extend MentionsExtractor
14 class SimpleMentionExtractor(MentionsExtractor):
15     # Implement predict function
16     def predict(self,
17                 docs: Iterable[Doc],
18                 batch_size=None) -> Iterable[Iterable[Span]]:
19         spans = [[Span(tok.idx, tok.idx + len(tok))
20                  for tok in doc if "s" in tok.text] for doc in docs]
21         return spans
22
23 # Create spacy pipeline
24 nlp = spacy.load("en_core_web_sm")
25 # Create config with the new custom
26 config = PipelineConfig(
27     mentions_extractor=SimpleMentionExtractor()
28 )
29 # Add zshot with custom component to the spacy pipeline
30 nlp.add_pipe("zshot", config=config, last=True)
31
32 text_acetamide = "CH202 is a chemical compound similar to Acetamide used in
33 International Business Machines Corporation (IBM)."
34 # Run the pipeline and print the result
35 doc = nlp(text_acetamide)
36 print([doc.text[mention.start:mention.end] for mention in doc._.mentions])
37 # -> ['is', 'similar', 'used', 'Business', 'Machines']

```

Listing 3: Example of creating a custom mentions extractor in Zshot.

Programmer: A person who writes computer programs.

Homemaker: A person who spends their time looking after a home and doing housework rather than being employed outside the home.

Michael Programmer and Maria Homemaker live in a small apartment

Figure 5: Example of bias of the SMXM linker model integrated in Zshot.

```

1 from zshot.utils.data_models import Entity
2 from zshot.linker import LinkerSMXM
3 from zshot import PipelineConfig, displacy
4 import spacy
5
6 nlp = spacy.blank('en')
7 config = PipelineConfig(
8     entities=[
9         Entity(name="company", description="The name of a company"),
10        Entity(name="location", description="A physical location"),
11        Entity(name="chemical compound", description="any substance composed of
12        identical molecules consisting of atoms of two or more chemical elements."),
13    ],
14    linker=LinkerSMXM()
15)
16 nlp.add_pipe("zshot", config=config, last=True)
17 text_acetamide = "CH2O2 is a chemical compound similar to Acetamide used in
18 International Business Machines Corporation (IBM)"
19
20 doc = nlp(text_acetamide)
21 displacy.render(doc, style="ent")

```

Listing 4: Example of annotating a sentence in ZShot.

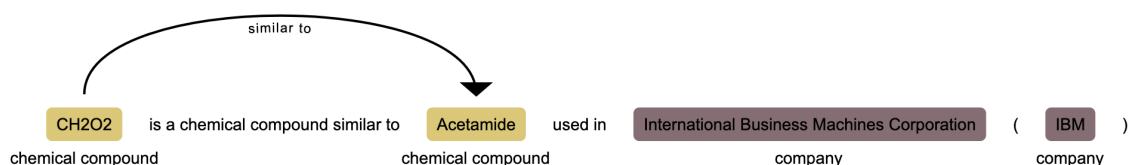


Figure 6: Example of sentence annotation with Zshot. Code of Listing 4 was used to generate this example.

Metric	SMXM ontotest-test	TARS ontotest-test
overall_precision_micro	22.12%	34.87%
overall_recall_micro	50.47%	48.22%
overall_f1_micro	30.76%	40.47%
overall_precision_macro	20.96%	28.31%
overall_recall_macro	48.15%	33.64%
overall_f1_macro	28.12%	28.29%
overall_accuracy	86.36%	90.87%
total_time_in_seconds	1811.5927	613.5401
samples_per_second	0.2352	0.6943
latency_in_seconds	4.2526	1.4402

Table 2: Full result of ZShot for evaluation. This experiment was performed on a MacBook Pro with an Intel-i7, 64Gb of RAM and no GPU. Results and comparison are only for illustration, as models have not been trained over the same data.

```

1 # import necessary libraries and define a spacy pipeline
2 import spacy
3 from zshot import PipelineConfig
4 from zshot.linker import LinkerSMXM, LinkerTARS
5 from zshot.linker.linker_ensemble import LinkerEnsemble
6 from zshot.utils.data_models import Entity
7 from zshot import displacy
8 nlp = spacy.blank("en")
9
10 # a list of two different descriptions of the entity "fruits"
11 enhanced_descriptions = [[Entity(name="fruits", description="The sweet and
12     fleshy product of a tree or other plant."),
13     [Entity(name="fruits", description="Names of fruits such
14     as banana, oranges")]]
15
16 # a list of two different pretrained linkers
17 linkers = [LinkerSMXM(), LinkerTARS()]
18
19 # Define an ensemble linker with the given enhanced descriptions and pretrained
20 linker models.
21 # The provided threshold=0.25 means that at least 1 pipeline out of 4 votes for
22 an output span.
23 # We have overall 4 pipelines in the ensemble (2 enhanced descriptions times 2
24 linkers).
25 ensemble=LinkerEnsemble(enhance_entities=enhanced_descriptions, linkers=linkers,
26     threshold=0.25)
27
28 # add the ensemble to the spacy NLP pipeline
29 nlp.add_pipe("zshot", config=PipelineConfig(linker=ensemble), last=True)
30
31 # annotate a piece of text
32 doc = nlp('Apples or oranges have a lot of vitamin C.')
33
34 # Visualize the result
35 displacy.render(doc)

```

Listing 5: Python example of making an ensemble of pipelines with two linkers and two different descriptions of the entity *fruits*.