Unsupervised Term Extraction for Highly Technical Domains

Francesco Fusco
IBM Research
ffu@zurich.ibm.com

Peter Staar
IBM Research
taa@zurich.ibm.com

Diego Antognini
IBM Research
Diego.Antognini@ibm.com

Abstract

Term extraction is an information extraction task at the root of knowledge discovery platforms. Developing term extractors that are able to generalize across very diverse and potentially highly technical domains is challenging, as annotations for domains requiring in-depth expertise are scarce and expensive to obtain. In this paper, we describe the term extraction subsystem of a commercial knowledge discovery platform that targets highly technical fields such as pharma, medical, and material science. To be able to generalize across domains, we introduce a *fully unsupervised* annotator (UA). It extracts terms by combining novel morphological signals from sub-word tokenization with term-to-topic and intra-term similarity metrics, computed using general-domain pre-trained sentence-encoders. The annotator is used to implement a *weakly-supervised setup*, where transformer-models are fine-tuned (or pre-trained) over the training data *generated* by running the UA over large unlabeled corpora. Our experiments demonstrate that our setup can improve the predictive performance *while decreasing* the inference latency on both CPUs and GPUs. Our annotators provide a very competitive baseline for all the cases where annotations are not available.

1 Introduction

Automated Term Extraction (ATE) is the task of extracting terminology from domain-specific corpora. Term extraction is the most important information extraction task for knowledge discovery systems – whose aim is to create structured knowledge from unstructured text – because domain specific terms are the linguistic representation of domain-specific concepts. To be of use in knowledge discovery systems (e.g., SAGA (Ilyas et al., 2022), DeepSearch (Dognin et al., 2020)) the term extraction has to identify individual *mentions* of terms to enable downstream components (i.e., the entity linker) to use not only the terms, but also their surrounding context. Unlike other applications of term extraction, such as text classification, where it is sufficient to extract representative terms for entire documents or even use generative approaches, term extraction in knowledge discovery systems has to be approached as a sequence tagging task.

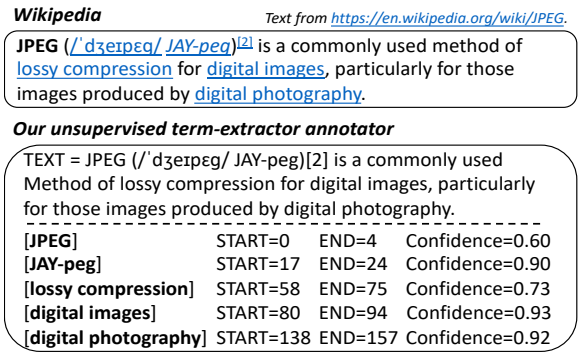

Figure 1: Our term extractor identifies the same mentions as Wikipedia without *relying on annotated data*.

The largest challenges for term extraction systems, when used for knowledge discovery, are generalization across domains and lack of annotated data. In fact, commercial knowledge discovery platforms are typically required to process large corpora targeting very diverse and often highly technical domains. Organizing annotation campaigns for such vertical domains is a costly process as it requires highly specialized domain experts. An additional challenge for such platforms are the computational requirements, which must be accounted for when developing technologies required to sift through very large and often proprietary corpora.

In this work, we describe an effective term extraction approach used in a commercial knowledge discovery platform[1] to extract *Wikipedia-like concepts*[2] from text (see Figure 1). Our approach does

[1] https://ds4sd.github.io.

[2] The linking from words to Wikilinks is done manually on Wikipedia, see https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking for more details.

Proceedings of EMNLP 2022 Industry Track, pages 1–8
December 9–11, 2020.

not require any human annotation, offers the flexibility to select the right trade-off between accuracy and inference latency, and enables the deployment of lightweight models running entirely on CPUs.

At its core, our approach is a *weakly supervised* setup (see Figure 2), where transformer models are fine-tuned (or even entirely pre-trained) using the *weak labels* generated by a *fully unsupervised* term annotator. The unsupervised annotator (UA) combines novel morphological and semantic signals to tag sequences of text corresponding to domain-specific terminology. In fact, in addition to part-of-speech tagging to identify candidate terms, the UA exploits sub-word tokenization techniques – commonly used in language models to highlight words that are outside of the common vocabulary – to indirectly measure the morphological complexity of a word based on its sub-tokens. To the best of our knowledge, this is the first work relying on sub-word tokenization units in the context of term extraction. To prune the candidate set of terms the annotator uses two semantic metrics as thresholds: the *topic-score* and a novel *specificity score* that are both computed using representations from sentence encoders. The unsupervised annotator, combined with the two-stage weakly supervised setup, makes our approach particularly attractive for practical industrial setups because computationally intensive techniques used by the unsupervised annotator are not paid at inference time. Therefore, one can improve the annotation quality by using more expensive techniques (e.g., entity linking to external knowledge bases), without adding costs at inference time. The two main contributions of this paper are summarized as follows:

1. We extract a novel morphology signal from subword-unit tokenization and we introduce a new metric called the *specificity score*. Upon those signals, we build an unsupervised term-extractor that offers competitive results when no annotation is available.

2. We show that by fine-tuning transformer models over the weak labels produced by the unsupervised term extractor we decrease the latency and improve the prediction quality.

2 Related work

Automated Term Extraction (ATE) is a natural language processing task that has been the subject of many research studies (Buitelaar et al., 2005; Lossio-Ventura et al., 2016; Zhang et al., 2018; Ma et al., 2019; Šajatović et al., 2019). What we describe in this work is an effective term extraction approach that is fully unsupervised and also offers the flexibility and modularity to deploy and easily maintain systems in production.

ATE should not be confused with keyphrase extraction (Firoozeh et al., 2020; Mahata et al., 2018; Bennani-Smires et al., 2018) and keyphrase generation (Wu et al., 2022; Chen et al., 2020), which have the goal of extracting, or generating, key phrases that best describe a given free text document. Keyphrases can be seen as a set of tags associated to a document. In the context of keyphrase extraction, sentence embedders have been used in the literature, such as in EmbedRank (Bennani-Smires et al., 2018) and Key2Vec (Mahata et al., 2018). In our work, we also rely on sentence encoders, but we use them to generate training data for sequence tagging. Therefore, we do not rely on sentence encoders at runtime to extract terminology from text, enabling the creation of lower latency systems.

To capture complex morphological structures we use word segmentation techniques. Word segmentation algorithms such as Byte-Pair Encoding (Sennrich et al., 2016), word-piece (Schuster and Nakajima, 2012), and unigram language modeling (Kudo, 2018) have been introduced to avoid the problem of out-of-vocabulary words and, more in general, to reduce the number of distinct symbols that sequence models for natural language processing have to process. To the best of our knowledge, we are the first to use the subword-unit tokenization as a signal to extract technical terms from text.

Our approach builds on the notion of specificity to find terminology. While there are multiple research works (Caraballo and Charniak, 1999; Ryu and Choi, 2006) highlighting the importance of specificity, to the best of our knowledge, this is the first work using the notion of specificity to extract terminology from text.

3 The approach

Figure 2 depicts our *weakly supervised* setup. Starting from a raw text corpus and no labels, our training workflow produces an efficient sequence tagging model, based on the transformer architecture, which effectively implements the term extraction. At the core of the *weak labels* there is a fully unsupervised component, called the *Unsupervised*

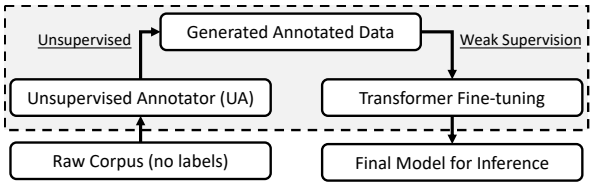

Figure 2: Our training workflow consists of 1) generating training data from raw unlabeled text using our Unsupervised Annotator, and 2) fine-tuning a transformer-based model or any sequence tagging model.

Annotator (UA), which, given the raw corpus, produces a training dataset for sequence labeling. The resulting dataset is used to train (or fine-tune) a sequence model that represents the final model for term annotation used at inference time. Pre-trained transformer-based models clearly represent a valid alternative to implement such sequence models. Moreover, we can avoid pre-training since the UA potentially generates a large amount of training data.

From the software engineering standpoint, this setup is extremely attractive as it makes the architecture of the term extraction subsystem modular and very flexible. The modularity comes from decoupling the inference component and the unsupervised annotator (UA). The unsupervised annotator can be enhanced with additional and more computationally demanding subcomponents (e.g., an entity linker to an external knowledge base), without increasing the final inference latency observed by the user. This modularity enables domain customization with proprietary data (and systems), which might be available for specific domains or customers. Since the integration between the Unsupervised Annotator and the inferencing component is achieved via data (i.e., the training samples for sequence tagging expressed in IOB format) the approach enables the smooth transition between a fully unsupervised setup and a setup where manual annotations augment the ones obtained via the UA. In practice, in realistic deployments, the unsupervised annotator is used to *boostrap* the term extraction subsystem, while domain specific annotations are added over time by organizing annotation campaigns or by collecting labels through the interactions of the users with the knowledge discovery platform.

Having a dedicated component for inferencing, which is independent from the UA, gives the flexibility to select the right trade-off in terms of accuracy, inference latency, deployment costs, and inferencing infrastructure. This choice is completely independent from the Unsupervised Annotator, which can be independently improved without taking care of inference latency. Since the inference component can be built around off-the-shelf transformer-based models, one can fully leverage the optimizations available in modern commercial offerings for inferencing services (e.g., Amazon Sagemaker, HuggingFace Infinity). As Transformer-based models are frequently used for multiple tasks (e.g., classification, NER, QA) within a knowledge discovery platform, this often corresponds to having a very homogeneous inferencing infrastructure in production. However, given that the UA can potentially generate a large amount of training samples, large pre-trained models are not a necessity, and even alternative architectures such as pQRNN (Kaliamoorthi et al., 2021) or pNLP-Mixer (Fusco et al., 2022) can be used.

3.1 Unsupervised annotator

Our unsupervised annotator is responsible for providing accuracy in potentially unseen domains *without* any training data, as depicted in Figure 1. It achieves this goal by using a greedy approach that processes each sentence of a raw corpus using the following steps:

1. Extract multiword expression candidates. Using the part-of-speech tags we extract multiword expression candidates, consisting of sequences of zero or more adjectives (ADJ) followed by nouns (NOUN) or proper nouns (PROPNs) sequences. This chunking step allows us to identify term candidates expressed via multiword expressions.

2. Filter candidates by specificity or topic score. Once the candidate terms, represented as multiword expression, are identified, a pruning step is responsible for filtering out multiword expressions using two semantic scores: the *topic score* and the *specificity score*. To compute those scores, we rely on pre-trained sentence encoders to extract embeddings from text.

■ **Topic score.** The topic score captures the similarity, topic-wise, between a candidate and the sentence containing it. It is computed as the cosine similarity between the embedding vector of the multiword expression and the embedding vector of the sentence containing it.

■ **Specificity score (SP).** This is the mean of the pairwise distance, in the embedding space, between the multiword expressions and all the other word or

multiword expression in the context. Specifically, given a multiword mw, and the word or multiword expression $w_1, ..., w_k$ in its context, we define the specificity score SP as:

$$SP(mw) = \frac{\sum_{i=1}^{k} dist(w_i, mw)}{k}, \qquad (1)$$

where $dist(w_i, w_j)$ is the cosine-similarity between the embedding vectors of w_i and w_j. Multiword expressions with a higher score correspond to more specific terms.

Multiword expressions with a specificity or topic score below a certain threshold can be filtered out. Both scores rely on high-quality sentence encoders. In our implementation we use the pretrained sentence encoders described in Reimers and Gurevych (2019), but other sentence encoders can be used as a drop-in replacement.

3. Upgrade single nouns according to morphological features. At this stage, we could have nouns that are not part of any multiword expressions, but still relevant. We deal with those cases separately. For each of those nouns, we have to decide whether to extract them as terms or not. To do so, we use morphological features. First, we check if the lemma of the noun is the same as any of the heads of the multiword expressions. If that is the case, we upgrade the noun to term. Otherwise, we segment the word using a subword-unit segmentation algorithm and a vocabulary trained over a large general purpose corpus. Subword-unit tokenizers have been introduced to enable the representation of any text as a combination of subword units, with the idea that the most frequent words can be represented by a small number of subword units, eventually just one for very common words as in case for stopwords. For example, the word *"sun"*, will have its own entry in the dictionary of subword units, while the word *"paracetamol"* will be represented as the sequence of the following subword units:[*"para", "##ce", "##tam", "##ol"*]. Not suprisingly, the number of subword units required to represent a word in a subword-unit tokenization regime is a very strong morphological signal, which we use as an *indirect measure* of the morphological "complexity", and is extremely cheap to compute. In our implementation, we simply promote as terms all the nouns with a number of sub-tokens higher than a threshold (4 in our case). We use the vocabulary of the BERT-base model from HuggingFace (Wolf et al., 2020) and the corresponding tokenizer.

	Sentence			Terms		
Corpus	Train	Dev	Test	Train	Dev	Test
ACL	828	276	280	2,574	898	930
GENIA	11,127	3,709	3,710	48,928	16,217	16,404
ScienceIE	2,516	417	876	6,067	1,052	1,885

Table 1: Number of sentences and terms in the train, dev, and test set for the datasetst used for evaluation.

4 Experiments

We now assess whether our approach can represent a *valid baseline* for term extraction in different technical domains when annotated data is *not available*. We aim to answer the following research questions:

- Does our Unsupervised Annotator generate a high-quality weakly-annotated dataset from a unlabeled general-domain corpus?
- Can we train models on the latter to lower the latency inference **and** increase the prediction performance at the same time?

4.1 Datasets

We use three common publicly available term extraction corpora: ACL RD-TEC 2.0 (QasemiZadeh and Schumann, 2016), GENIA (Kim et al., 2003), and ScienceIE (Augenstein et al., 2017). Each contains abstracts from scientific articles in different domains: natural language processing (ACL), medicine (GENIA), and computer science, material science, as well as physics (ScienceIE). All tokens are annotated using the IOB format (short for Inside, Out and Begin) (Ramshaw and Marcus, 1999). Since we are only interested in general term extraction, we did not use multiple class labels, even if provided in the respective dataset. We create random splits of train, dev, and test sets (60/20/20) for the ACL and GENIA datasets, and we use the pre-existing data splits for ScienceIE corpus.

In terms of preprocessing, we remove nested terms from the GENIA dataset, since the IOB tag set does not allow nested term extraction. For the ACL corpus, some samples have abstracts labeled by two annotators. In those cases, we selected the abstract from the first annotator. An overview of the datasets is given in Table 1.

Since our objective is to study the generalization of our approach, we need an *unlabeled* broad corpus from which our Unsupervised Annotator will annotate the text. Hence, we randomly sampled 500,000 sentences from abstracts from Semantic

	ACL			GENIA			ScienceIE	
Model (#Params)	exact F_1	partial F_1	Model (#Params)	exact F_1	partial F_1	Model (#Params)	exact F_1	partial F_1
BERT B (110M)	78.69	91.06	BERT B (110M)	70.13	88.19	BERT B (110M)	49.62	66.36
ELECTRA S (14M)	72.84	88.06	ELECTRA S (14M)	67.73	88.04	ELECTRA S (14M)	46.43	68.45
ELECTRA XS (7M)	50.40	71.61	ELECTRA XS (7M)	59.86	83.16	ELECTRA XS (7M)	27.17	51.10
UA (0)	49.95	74.56	UA (0)	45.65	77.16	UA (0)	39.75	64.29

Table 2: Results for the unsupervised annotator (UA) and transformer models fine-tuned on the **manually annotated** ACL, GENIA, and ScienceIE datasets, respectively. Without using any annotation, the UA performs similarly to ELECTRA XSmall and even better on the ScienceIE.

Scholar (SS). [3] We call our weakly annotated training set UA-SS. The training sets of the ACL, GENIA, and ScienceIE datasets are not used (unless specified).

4.2 Models

We use transformer-models, fine-tuned with manual annotations, as baselines. We employ pre-trained transformer models of different sizes: BERT-base (110M parameters) (Devlin et al., 2019), ELECTRA Small (14M parameters) (Clark et al., 2020), and ELECTRA XSmall (7M parameters).

Since our main goal is to compare the models to each other and across multiple corpora, we prioritize comparability across corpora over comparability with approaches from other studies.

4.3 Experimental settings

We use the pre-trained checkpoints of BERT-base and ELECTRA Small from HuggingFace (Wolf et al., 2020). We pre-train ELECTRA XSmall[4] from scratch using our Semantic Scholar dataset. During fine-tuning, we devoted a similar amount of GPU time to all the models. We pick the best-performing model in the dev set after 10 epochs

We implemented our Unsupervised Annotator using the POS tagger of SpaCy (Honnibal et al., 2020). To compute the specificity and similarity scores we use the sentence embedding model `distilbert-base-nli-mean-tokens` from the `sentence transformers`[5] library.

The specificity and similarity thresholds used to generate the training data over abstracts from Semantic Schoolar have been set to conservative values. We set the threshold for the specificity $T_{SP} = 0.05$ and the threshold for the similarity $T_{topic} = 0.1$. For the sub-word tokenization we rely on the tokenizer from BERT-base.

4.4 Results

In Table 2, we first compare the performance (expressed as exact and partial F_1 scores that count only exact or partial matches as true positives) of our fully *Unsupervised Annotator* to the performance obtained by fine-tuning transformer-based models with the manual annotations present in the original training sets. Without relying on any human annotation, our UA delivers comparable or even better results than the ELECTRA XSmall in ACL and ScienceIE, respectively. These results show that the UA represents a very *competitive baseline* for domains where annotations are not available.

Further, we are interested in understanding whether transformer-based models fine-tuned with human annotations can generalize across domains. We also evaluate if the availability of weakly supervised labels generated by our *Unsupervised Annotator* over a large and broad corpus (i.e., Semantic Scholar) could lead to models with higher generalization capabilities. In Table 3 we report the exact and partial F_1 scores for the ACL, GENIA, and ScienceIE datasets, and the transformer-based model fine-tuned with the output of our *Unsupervised Annotator* (UA-SS). This setup simulates the problem of bootstrapping an annotator for a specific domain for which in-domain human labels are not available.

On the ACL corpus, the UA-SS-based model clearly outperforms the GENIA-based and ScienceIE-based models. On the GENIA corpus, the UA-SS-based model and the ACL-based model perform equally well. On the ScienceIE corpus, all models perform equally with a slight tendency towards the GENIA-based model.

Overall, it can be said that the UA-SS-based approach is a valid starting point to bootstrap a

[3] www.semanticscholar.org/.

[4] We used 2 attention heads and 4 hidden layers, while using the same hidden dimension and similarly sized vocabulary.

[5] pypi.org/project/sentence-transformers/.

Model (#Params)	Fine-tuned on	ACL		GENIA		ScienceIE	
		exact F_1	partial F_1	exact F_1	partial F_1	exact F_1	partial F_1
BERT Base (110M)	UA-SS	**58.22**	**77.36**	**53.18**	79.38	46.79	66.59
	ACL	–	–	52.05	**82.49**	47.88	69.97
	GENIA	45.97	61.53	–	–	**48.50**	**69.84**
	ScienceIE	38.28	54.92	46.91	73.16	–	–
ELECTRA Small (14M)	UA-SS	**58.00**	**77.41**	**53.44**	80.01	44.68	65.58
	ACL	–	–	50.84	**81.33**	44.21	67.57
	GENIA	46.65	67.21	–	–	**45.79**	**68.83**
	ScienceIE	42.58	66.02	43.48	76.77	–	–
ELECTRA XSmall (7M)	UA-SS	**49.83**	**72.78**	**45.35**	**74.83**	**40.32**	**62.39**
	ACL	–	–	31.13	59.99	28.79	58.20
	GENIA	29.81	58.17	–	–	30.00	59.61
	ScienceIE	20.60	33.53	39.95	68.63	–	–

Table 3: Results for the generalization of multiple transformer models that are fine-tuned on the **weakly annotated** dataset based on the Semantic Scholar corpus (annotated with UA, denoted as UA-SS) and evaluated on the ACL, GENIA, and ScienceIE datasets, respectively. Transformer models fine-tuned using our automatically generated dataset perform better than their counterparts fine-tuned using the other datasets.

system in a no-resource scenario. Table 2 shows that the F1 score gap between models trained with in-domain manually annotated data and the UA-SS-based approach is lower for smaller models.

Now, we compare the *Unsupervised Annotator* with the models fine-tuned with its output to evaluate our two-step approach in terms of F1 score *and* inference latency. Figure 3 reports the average inference latency for models (fine-tuned with the UA-SS training data) over sentences from the ACL dataset with a batch of size 1 using a NVIDIA Tesla V100 and a single core of a Xeon E5-2690 v4 (similar trends on the other datasets). While the inference latency has similar orders of magnitude across models with GPU acceleration, the minimum inference time of 26.6 ms can be obtained on a *single CPU core* using the ELECTRA XSmall model. Therefore, our approach is particularly attractive in all cases where inference accelerators (e.g., GPUs) are not available. Additionally, the results highlight that by fine-tuning over the output of the UA, the latency can be *reduced* by *4 to 10 times*, while providing comparable or even better F1 scores. Having the option to generate a large amount of training data for fine-tuning is an extremely useful property that enables the creation of very small models offering low inference times even without using GPU acceleration.

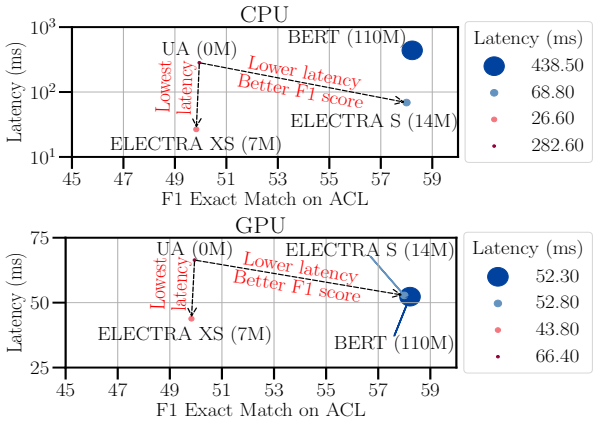

Figure 3: Average inference latency on CPU (top) and GPU (bottom) on the ACL dataset. We note in parenthesis the number of trainable parameters of the models. By fine-tuning over the output of the UA, we achieve lower latency and higher F1 scores. The lowest inference latency, 26.6 ms, is achieved on CPU.

4.5 Lessons learned

In this work, we have demonstrated that, while the value of in-domain labels is without any doubt the best way to increase predictive quality, fully unsupervised approaches are often the only viable option to bootstrap a term extractor that has to generalize across very diverse domains. Additionally, while the practicality of ML solutions is often underestimated, we have shown that having a modular system can not only provide greater flexibility in deployments, but can also allow to boost time predictive performance and inference latency at the same.

5 Conclusion

In this paper, we described an effective term extraction approach that uses a *fully unsupervised*

annotator to generate training data to fine-tune transformer models. This approach reduces the inference time of the unsupervised annotator, without decreasing its performance, and allows the flexibility to pick the right trade-off between latency and F1 score. The latency-optimized models are less than 30 Megabytes in size, provide inference latencies lower then 30 ms even *without* GPUs, while exhibiting a competitive F1 score compared to the models fine-tuned with manually annotated data.

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.

Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*.

Sharon A. Caraballo and Eugene Charniak. 1999. Determining the specificity of nouns from text. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pierre Dognin, Igor Melnyk, Inkit Padhi, Cicero Nogueira dos Santos, and Payel Das. 2020. DualTKB: A Dual Learning Bridge between Text and Knowledge Base. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8605–8616, Online. Association for Computational Linguistics.

Nazanin Firoozeh, Adeline Nazarenko, Fabrice Alizon, and Béatrice Daille. 2020. Keyword extraction: Issues and methods. *Natural Language Engineering*, 26(3):259–291.

Francesco Fusco, Damian Pascual, and Peter Staar. 2022. pNLP-Mixer: an Efficient all-MLP Architecture for Language. *arXiv preprint arXiv:2202.04350*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, and Mohamed Soliman. 2022. Saga: A platform for continuous construction and serving of knowledge at scale. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD/PODS '22, page 2259–2272, New York, NY, USA. Association for Computing Machinery.

Prabhu Kaliamoorthi, Aditya Siddhant, Edward Li, and Melvin Johnson. 2021. Distilling large language models into tiny and effective students using pqrnn. *CoRR*, abs/2101.08890.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Juan Antonio Lossio-Ventura, Clément Jonquet, Mathieu Roche, and Maguelonne Teisseire. 2016. Biomedical term extraction: overview and a new methodology. *Information Retrieval Journal*, 19(1-2):59–99.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring sequence-to-sequence learning in aspect term extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3538–3547, Florence, Italy. Association for Computational Linguistics.

Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. 2018. Key2Vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 634–639, New Orleans, Louisiana. Association for Computational Linguistics.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. The acl rd-tec 2.0: A language resource for evaluating term extraction and entity recognition methods. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1862–1868.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

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

Pum-Mo Ryu and Key-Sun Choi. 2006. Taxonomy learning using term specificity and similarity. In *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 41–48, Sydney, Australia. Association for Computational Linguistics.

Antonio Šajatović, Maja Buljan, Jan Šnajder, and Bojana Dalbelo Bašić. 2019. Evaluating automatic term extraction methods on individual documents. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 149–154, Florence, Italy. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Di Wu, Wasi Uddin Ahmad, Sunipa Dev, and Kai-Wei Chang. 2022. Representation learning for resource-constrained keyphrase generation. *CoRR*, abs/2203.08118.

Ziqi Zhang, Johann Petrak, and Diana Maynard. 2018. Adapted textrank for term extraction: A generic method of improving automatic term extraction algorithms. *Procedia Computer Science*, 137:102 – 108. Proceedings of the 14th International Conference on Semantic Systems 10th – 13th of September 2018 Vienna, Austria.