SEAGLE: A Platform for Comparative Evaluation of Semantic Encoders for Information Retrieval

Fabian David Schmidt[*], **Markus Dietsche**[*], **Simone Paolo Ponzetto and Goran Glavaš**

Data and Web Science Group
University of Mannheim
fabian.david.schmidt@hotmail.de
dietsche.markus@googlemail.com
{simone, goran}@informatik.uni-mannheim.de

Abstract

We introduce SEAGLE,[1] a platform for comparative evaluation of semantic text encoding models on information retrieval (IR) tasks. SEAGLE implements (1) word embedding aggregators, which represent texts as algebraic aggregations of pretrained word embeddings and (2) pretrained semantic encoders, and allows for their comparative evaluation on arbitrary (monolingual and cross-lingual) IR collections. We benchmark SEAGLE's models on monolingual document retrieval and cross-lingual sentence retrieval. SEAGLE functionality can be exploited via an easy-to-use web interface and its modular backend (micro-service architecture) can easily be extended with additional semantic search models.

1 Introduction and Motivation

Traditional IR models operate on lexical overlap and fail to identify relevance when documents and queries differently lexicalize concepts. Semantic search seeks to overcome such lexical mismatches between document and user queries (Li and Xu, 2014). Early approaches to semantic search relied on external resources like WordNet (Moldovan and Mihalcea, 2000) and Wikipedia (Strube and Ponzetto, 2006), suffering from the resource's limited coverage. More recent semantic search systems (Vulić and Moens, 2015; Litschko et al., 2018; Nogueira and Cho, 2019) remedy for those coverage issues by encoding text using distributional word vectors (i.e., word embeddings) (Mikolov et al., 2013; Bojanowski et al., 2017) and neural text encoders (Devlin et al., 2018).

While there is a plethora of semantic text encoding models, there have been few attempts to empirically compare them on IR tasks. In this work, we aim to allow for such comparative evaluations on arbitrary IR test collections. We introduce SEAGLE, a platform for concurrent execution and comparative evaluation of semantic search models. SEAGLE implements most recently proposed (1) word embedding aggregation models (Arora et al., 2017; Rücklé et al., 2018; Yang et al., 2019; Zhelezniak et al., 2019) as well as (2) two pretraining-based text encoders (Gysel et al., 2017; Devlin et al., 2018) and allows users to evaluate them on arbitrary IR collections. Coupled with pretrained cross-lingual embedding spaces (Glavaš et al., 2019), SEAGLE also supports cross-lingual search. The platform's modular architecture based on micro-services makes it easy to extend it with additional semantic encoding models. SEAGLE is accessible via an easy-to-use web interface.

2 Semantic Representation Models

We first describe SEAGLE's semantic encoders, belonging to two categories: word embedding aggregators and pre-trained text encoders.

2.1 Word Embedding Aggregators

Word embedding aggregators encode the text by aggregating pretrained d-dimensional embeddings of its words. Formally, a document matrix $V_d \in \mathbb{R}^{N \times d}$ sequentially stacks embeddings $\mathbf{t}_i \in \mathbb{R}^d$ corresponding to tokens t_i ($i \in \{1, \cdots, N\}$) of document d from the collection D. A weight w_i is computed for every token t_i and the contribution of the embedding $\mathbf{t}_i$ to the document representation $\mathbf{d} \in \mathbb{R}^d$ is scaled according to w_i.

Continuos Bag-of-Words (CBOW) simply averages the rows of the document embedding matrix V_d. Put differently, CBOW computes the simple average of the word embeddings, i.e., it assigns equal weights (w_i=$1/N$) to all embeddings.

Term Frequency-Inverse Document Frequency

[*]These authors contributed equally to this paper.

[1]SEAGLE is available on GitHub and demonstrated on YouTube.

Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations), pages 199–204
Hong Kong, China, November 3 – 7, 2019.

(TF-IDF) aggregator computes the weight w_i as the product of term t_i's relative frequency within the document d (TF) and the inverse of proportion of documents in the collection containing t_i (IDF). The assumption is that (1) more frequent words contribute more to the document's meaning, as do (2) the terms that are more specific to the document (i.e., do not occur in many other documents).

Smooth Inverse Frequency (SIF) conflates bias-adjusted weighted word embeddings to generate document embeddings (Arora et al., 2017):

$$\mathbf{d} = \frac{1}{N} \sum_{t_i \in d} \frac{a}{a + p(t_i|D)} \mathbf{t}_i \quad (1)$$

Weight w_i of a term t_i is a smoothed inverse of the probability of t_i under the unigram language model built from the whole collection D, with a being the smoothing factor. In the next step, common component removal (CCR) is applied to every document vector $\mathbf{d}$: let $X \in \mathbb{R}^{d \times |D|}$ be the matrix obtained by stacking vectors of all collection documents as columns. Each document vector $\mathbf{d}$ is then replaced with $\mathbf{d} - \mathbf{u}\mathbf{u}^\top \mathbf{d}$, where $\mathbf{u}$ is the left singular vector of X. CCR aims to eliminate the similarities between document vectors that originate from syntactic rather than semantic similarities.

Concatenated Power Means (CPM) generalizes the aggregation of word vectors to their chosen powers (Rücklé et al., 2018):

$$\mathbf{d}^p = \left(\frac{\mathbf{t}_1^p + \cdots + \mathbf{t}_n^p}{N}\right)^{\frac{1}{p}} \quad (2)$$

Choices for p constitute a hyperparameter and determine the dimensionality of the resulting document embedding, as $\mathbf{d}^p$ for different p are concatenated into a final document representation. Power means can reduce a set of vectors to a geometric mean ($p = 0$), arithmetic mean ($p = 1$), minimum ($p = -\infty$), and maximum ($p = \infty$). We concatenate the last three to generate the final document embedding, and then, following the original work (Rücklé et al., 2018), perform element-wise z-normalization over document vectors $\mathbf{d}$.

Geometric Embedding (GEM) weighs word embeddings $\mathbf{t_i}$ by summing their *novelty*, *significance*, and *uniqueness* scores (Yang et al., 2019) and correcting the resulting document vectors for (document-dependent) principal components via CCR. Let $W^i = [\mathbf{t}_{i-m}, \mathbf{t}_{i-1}, \cdots, \mathbf{t}_{i+m}, \mathbf{t}_i] \in \mathbb{R}^{d \times (2m+1)}$ be the contextual window of the token t_i with m neighbours.

The *novelty score* of t_i is computed as the normalized minimal distance from from $\mathbf{t}_i$ to the subspace spanned by the vectors of context words. The *significance score* captures the semantic alignment between the vectors $\mathbf{t}_i$ and the context W^i as the similarity between $\mathbf{t}_i$ and singular vectors of W^i obtained via SVD. Intuitively, a token is more significant the more it is aligned with the context's principal components. Lastly, the *uniqueness* score quantifies the alignment between the word vector and the principal directions computed on the whole collection. A token highly aligned with collection's principal components is an uninformative word and should receive a low weight.[2]

DynaMax-Jaccard (DJ) is a non-parametric similarity measure (Zhelezniak et al., 2019). The algorithm projects stacked word embeddings of a query $\mathbf{q} \in \mathbb{R}^{M \times d}$ and stacked word embeddings of a document $\mathbf{d} \in \mathbb{R}^{N \times d}$ into the shared space $U \in \mathbb{R}^{(M+N) \times d}$. Features for $\mathbf{q}$ and $\mathbf{d}$ are then max-pooled along the rows of projections, respectively. Feature generation tests the degree of membership of $\mathbf{q}$ and $\mathbf{d}$ in U and represents an extension of set theory to real-valued vectors. Accordingly, the fuzzy Jaccard index measures the similarity between query and document representations and is computed as follows: stacked features are min- and max-pooled over rows and the sum of minima over the sum of maxima yields the fuzzy Jaccard score.

2.2 Semantic Text Encoders

BERT (Devlin et al., 2018) is a general-purpose unsupervised pretraining model based on the Transformer architecture that can dynamically predict contextualized token vectors. We integrate `bert-as-a-service` (Xiao, 2018) to infer document embeddings: we average-pool stacked token representations in each of BERT's Transformer's layers (second to fourth from the last layer) and concatenate the resulting averaged representations of each layer to obtain a document embedding d. We then element-wise z-normalize $\mathbf{d}$. Because BERT's positional encoding limits the maximal sequence length, we dissect collection documents into 256-token segments (with 32 token overlap between adjacent segments). The final document embedding $\mathbf{d}$ is the average of 256-token segments' embeddings generated by BERT.

[2]For more details on GEM, we refer the reader to the original work (Yang et al., 2019).

Neural Vector Space Model (NVSM) jointly learns token and document representations (Gysel et al., 2017). Specifically, during training (inference), relevant queries are modeled as CBOW embeddings of n-grams (query) sampled from the source document and then mapped via a learned transformation onto the document space. NVSM then learns its parameters (word and document embeddings; mapping), such that the mapped n-grams (queries) are most similar to the respective document representation. The model accounts for the lack of positive supervision (the number of relevance judgments is typically rather limited) via negative sampling and a contrastive maximum likelihood loss. For more details on NVSM, we refer the reader to the original work (Gysel et al., 2017).

3 Benchmarking Semantic Models

We benchmark the above semantic search models on two retrieval tasks: (1) monolingual document retrieval on the LATimes test collection (112.082 documents, 114 queries, and 2.094 positive relevance judgments) (2) cross-lingual sentence retrieval on the subset of Europarl's sentence-aligned corpora: we compile 5.000 aligned sentence translations between English (EN), Italian (IT), German (DE), and Finnish (FI). Following the specificities of each collection, we measure performance on LATimes collection in terms of NDCG@100, MAP@1000, and Precision@10; and in terms of MRR and Hits@1, 5, 10 for Europarl. We additionally evaluate BM25, a robust probabilistic retrieval model (Robertson et al., 2009) as a baseline for the monolingual document retrieval. BM25 captures only lexical overlaps between documents queries, i.e., it cannot capture any semantic alignment not originating from shared terms.

3.1 Experimental Setup

All required corpus statistics (e.g., IDF or common components) are computed offline on the document collection as a preprocessing step. For all aggregation based methods (see §2.1) we employ 300-dimensional fastText embeddings (Bojanowski et al., 2017), pretrained on Wikipedia.[3] For cross-lingual sentence retrieval, we first induce the shared bilingual word embedding spaces by projecting the EN vectors to the monolingual space of the target language (IT, DE, or FI). To this end, following (Glavaš et al., 2019), we use 5K automatically obtained word translation pairs to induce the projection matrices by solving the Procrustes problem. We infer contextualized embeddings with pretrained BERT models using `Bert-Large, Uncased (Whole Word Masking)` and `Bert-Base, Multilingual Cased`[4] for LATimes and Europarl, respectively. Except for BERT, we lowercase and tokenize text using BlingFire.[5] For NVSM, we trim the vocabulary to 60k most frequent non-stop words and learn 512 dimensional word and document embeddings with a *tanh* activation mapping on sampled n-grams of length 16 with 10 contrastive examples.

3.2 Results

Table 1 summarizes the monolingual document retrieval and cross-lingual sentence retrieval results. NVSM yields the best retrieval performance on the monolingual document retrieval task, suggesting that reliable word and document representations can be learned from regular-size retrieval collections. Among the embedding aggregation models, SIF seems to display the best performance. The fact that BM25, a semantically unaware baseline, outperforms all semantic models on document retrieval is discouraging. However, this may merely be an artifact of the LATimes dataset: out of 2.094 relevances, the document contains some (all) query terms in 1.975 (647) cases.

The cross-lingual sentence retrieval shows that all semantic search models outperform the simple word embedding averaging. Overall, DynaMax-Jaccard (DJ) yields the strongest performance, only trailing SIF on the EN-FI evaluation. In EN-FI scenario the common component removal (CCR) step included in SIF strongly boosts the performance (SIF weighting without CCR yields merely 41.1% MRR). Our cross-lingual sentence retrieval based on the pre-trained multilingual BERT model exhibits strong performance across all three language pairs – on EN-IT and EN-DE it lags behind DJ by a small margin and substantially outperforms all other aggregators; on EN-FI it outperforms DJ but falls behind the surprisingly effective SIF (with CCR). BERT's and DJ's effectiveness, however, significantly drop in the monolingual document

[3] Available at https://fasttext.cc/docs/en/pretrained-vectors.html

[4] https://github.com/google-research/bert

[5] https://github.com/Microsoft/BlingFire

	LATimes			Europarl											
	EN-EN			EN-IT				EN-DE				EN-FI			
Model	NDCG	MAP	P@10	MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10
CBOW	28.4	18.8	14.6	59.4	52.9	66.4	71.1	55.4	48.7	62.7	67.5	38.3	30.7	46.6	52.8
TF-IDF	29.8	21.8	18.1	68.4	63.0	74.5	78.4	63.6	57.6	70.2	74.8	42.2	34.6	50.3	57.6
SIF	34.1	28.8	25.9	86.4	82.4	91.2	93.1	80.2	75.3	86.0	88.8	61.8	54.0	70.6	75.9
CPM	30.0	21.5	17.4	78.9	73.3	85.6	88.6	72.1	65.6	79.3	83.6	42.7	33.6	52.2	59.6
GEM	26.8	22.2	19.2	81.0	76.5	86.7	89.4	74.7	68.6	82.0	85.9	42.5	34.4	50.7	58.0
DJ	23.0	14.3	11.6	93.7	92.1	95.6	96.6	89.0	86.3	92.2	94.0	51.1	42.3	61.3	67.6
NVSM	31.8	34.9	29.7	–	–	–	–	–	–	–	–	–	–	–	–
BERT	20.2	12.4	11.5	87.3	84.4	90.6	92.7	87.4	84.5	90.8	92.7	56.5	49.2	64.9	71.0
BM25	38.0	42.0	32.5	–	–	–	–	–	–	–	–	–	–	–	–

Table 1: Results of the comparative evaluation of semantic search models on: (1) monolingual document retrieval (LATimes); metrics: NDCG@100, MAP@1000 and Precision@10; and (2) cross-lingual sentence retrieval (Europarl, 5K sentence pairs, EN-DE, EN-IT, and EN-FI); metrics: MRR and Hits@{1, 5, 10}.

retrieval setup where the models need to encode much longer documents.

Lastly, the complexity and runtime of the evaluated models should not be ignored. As a rule of thumb, CBOW, TF-IDF, SIF and CPM, are quite efficient and resource-light, whereas the remaining algorithms become increasingly resource-demanding and decreasingly efficient – from GEM, and DJ, which require embedding aggregation over the whole collection, over predicting vectors with BERT (one inference for each max. length token segment), to NVSM, for which any collection change warrants model retraining.

4 Architecture & Interface

SEAGLE is implemented as an application based on micro-services, consisting of a web client (see figure 1) for configuration and search and network daemons (see figure 2), one for each semantic search model. Such a modularized architecture facilitates the implementation and addition of new semantic search models (as new daemons).

4.1 SEAGLE's Architecture

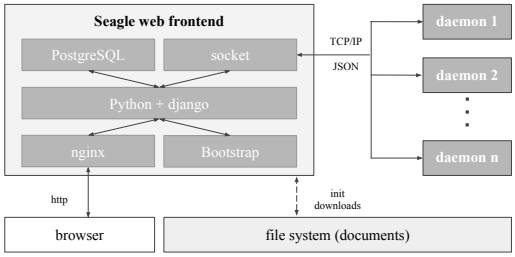

Figure 1: web front end components

Seagle is a Python application, based on the Django[6] framework for web development. It indexes document collections (i.e., all document representations required by semantic search models) within a PostgreSQL database.[7] It runs on a nginx[8] web-server and utilises Bootstrap[9] to ensure responsiveness.

Communication between the web application and the network daemons implementing the semantic search models is conducted through web sockets via TCP/IP and a JSON API. TCP was chosen over UDP so network daemons themselves can have a increased degree of control about how many searches and initializations are executed in parallel. This becomes relevant in case of large document collections and computationally intensive algorithms.

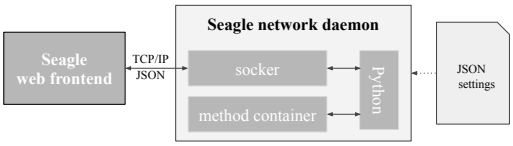

Figure 2: network daemon layout

Network Daemon Server. On the server side SEAGLE's Python-based network daemons contain the actual semantic search models. They are supplied with the document collection via the front-end and do not need to access the file system themselves. Each daemon comes with a template which implements the API and loads its settings from a local JSON file (see figure 2) containing setup information (e.g., the port on which it should run). To implement new functionality the template needs

[6] https://www.djangoproject.com/
[7] https://www.postgresql.org/
[8] https://www.nginx.com/
[9] https://getbootstrap.com/

to be extended with an actual method, containing the initialization function (i.e., a specification of the document indexing procedure for a particular retrieval model) and the search function (i.e., the specification of the ranking function for a particular model). In case multiple network daemons running on the same machine, they should all run on different ports.

Backend Deployment. SEAGLE daemons comprise a modular backend in which all semantic embedding models reuse I/O and evaluation functions, allowing for easy integration of additional search methods. Moreover, for efficiency, we implement all encoders using matrix and vector operations from Numpy.[10] While SEAGLE offers an easy-to-use web interface, semantic search models can be executed and evaluated from the command line.

A major benefit of the micro service architecture is its distributability on multiple machines. The main reason for multi-machine deployment of SEAGLE's backend is the computational complexity of some search models. Considering that for some models it might take hours or days to initialize (i.e., index) large document collections, the initialization process can be significantly reduced by deploying computation intensive models to different machines and running them in parallel.

4.2 SEAGLE Interface

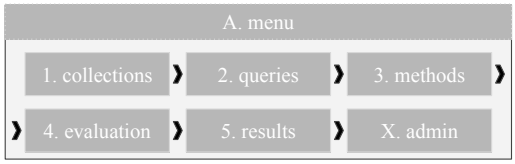

Figure 3: Components of SEAGLE's web interface

SEAGLE's web interface allows users, without programming and IR knowledge, to easily index document collections, select desired semantic search models and conduct comparative retrieval evaluations. Its core functionality is wrapped up in a easy to navigate one page layout and bundles 5 components (see figure 3):

1. Collections: Allows selection of one or multiple collections of documents on which methods are going to be evaluated.

[10] SEAGLE's reliance on Numpy routines requires a carefully tuned Numpy installation linked to fast linear algebra libraries.

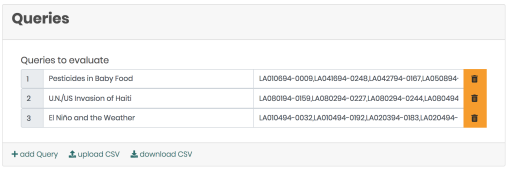

Figure 4: SEAGLE's queries component

2. Queries (see figure 4): 2.1. Manually add and remove queries (and their respective document relevance annotations); 2.2. Bundle CSV import and export & download of arbitrary number of queries with relevance annotations.

3. Methods: Select which semantic search methods to execute and evaluate.

4. Evaluation (see figure 5): 4.1. Change evaluation parameters, e.g. include or exclude evaluation metrics or specify the number of results to be shown for each method. 4.2. Contains a summary of collections, queries and methods, which are going to be evaluated along with the button triggering the evaluation process.

5. Results: 5.1. Bar charts of evaluation metrics, (e.g. MAP) or execution time per method. 5.2. A query explorer, allowing real-time exploration of executed conducted queries and top-ranked results by selected search models. (see figure 6)

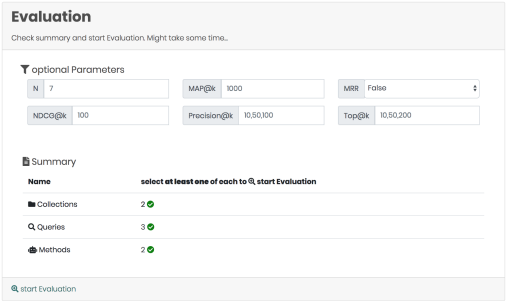

Figure 5: SEAGLE's evaluation component

Figure 6: SEAGLE's evaluation component

Additionally, there is a quick-link component for administration (X. admin), enabling quick setup and modifications of SEAGLE's backend.

5 Conclusion

We presented SEAGLE, a platform implementing a number of strong baselines for semantic IR and allowing for their comparative evaluation on arbitrary test collections. We benchmark the implemented semantic search models on monolingual document retrieval and cross-lingual sentence retrieval tasks, offering insights into their comparative advantages and shortcomings. Our benchmarking results indicate that there is no single best-performing semantic search model for all settings and that the users must consider various factors when selecting the best model for their retrieval task.

SEAGLE offers a satisfying out-of-the-box solution for fast benchmarking of semantic retrieval models on arbitrary collections. The platform's web interface allows the user to effortlessly index document collections, select semantic search models and their hyperparameter setup, comparatively evaluate selected models and finally visualize the results for manual inspection. SEAGLE's modular architecture (based on network daemon templates) allows for fast implementation of new search models and their inclusion into comparative evaluations in a plug-and-play fashion.

Acknowledgments

We thank Leon Schüller and Siying Liu, who contributed to the original student project on which SEAGLE is based on. Additionally, we thank Hans-Peter Zorn from inovex GmbH for his feedback over the course of the same student project.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. *arXiv preprint arXiv:1902.00508*.

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2017. Neural vector spaces for unsupervised information retrieval. *CoRR*, abs/1708.02702.

Hang Li and Jun Xu. 2014. Semantic matching in search. *Found. Trends Inf. Retr.*, 7(5):343–469.

Robert Litschko, Goran Glavaš, Simone Paolo Ponzetto, and Ivan Vulić. 2018. Unsupervised cross-lingual information retrieval using monolingual data only. In *SIGIR*, pages 1253–1256.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Dan I Moldovan and Rada Mihalcea. 2000. Using wordnet and lexical operators to improve internet searches. *IEEE Internet Computing*, 4(1):34–43.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated power mean embeddings as universal cross-lingual sentence representations. *arXiv*.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, volume 6, pages 1419–1424.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *SIGIR*, pages 363–372.

Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

Ziyi Yang, Chenguang Zhu, and Weizhu Chen. 2019. Zero-training sentence embedding via orthogonal basis.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. 2019. Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In *International Conference on Learning Representations*.