A Neural Autoencoder Approach for Document Ranking and Query Refinement in Pharmacogenomic Information Retrieval

Jonas Pfeiffer[1,2] **Samuel Broscheit**[1] **Rainer Gemulla**[1] **Mathias Göschl**[2]
[1]University of Mannheim, Mannheim, Germany
[2]Molecular Health GmbH, Heidelberg, Germany
{lastname}@informatik.uni-mannheim.de
Mathias.Goeschl@molecularhealth.com
Jonas.Pfeiffer.90@Gmail.com

Abstract

In this study, we investigate learning-to-rank and query refinement approaches for information retrieval in the pharmacogenomic domain. The goal is to improve the information retrieval process of biomedical curators, who manually build knowledge bases for personalized medicine. We study how to exploit the relationships between genes, variants, drugs, diseases and outcomes as features for document ranking and query refinement. For a supervised approach, we are faced with a small amount of annotated data and a large amount of unannotated data. Therefore, we explore ways to use a neural document auto-encoder in a semi-supervised approach. We show that a combination of established algorithms, feature-engineering and a neural auto-encoder model yield promising results in this setting.

1 Introduction

Personalized medicine strives to relate genomic detail to patient phenotypic conditions (such as disease, adverse reactions to treatment) and to assess the effectiveness of available treatment options (Brunicardi et al., 2011). For computer-assisted decision making, knowledge bases need to be compiled from published scientific evidence. They describe *biomarker* relationships between key entity types: *Disease*, *Protein/Gene*, *Variant/Mutation*, *Drug*, and *Patient Outcome (Outcome)* (Manolio, 2010). While automated information extraction has been applied to simple relationships — such as *Drug-Drug* (Asada et al., 2017) or *Protein-Protein* (Peng and Lu, 2017); (Peng et al., 2015); (Li et al., 2017) interaction — with adequate precision and recall, clinically actionable biomarkers need to satisfy rigorous quality criteria set by physicians and therefore call upon manual data curation by domain experts.

To ascertain the timeliness of information, curators are faced with the labor-intensive task to identify relevant articles in a steadily growing flow of publications (Lee et al., 2018). In our scenario, curators iteratively refine search queries in an electronic library, such as PubMed.[1] The information the curators search for, are biomarker-facts in the form of {*Gene(s) - Variant(s) - Drug(s) - Disease(s) - Outcome*}. For example, a curator starts with a query consisting of a single gene, e.g. $q_1 = \{PIK3CA\}$, and receives a set of documents D_1. After examining D_1, the curator identifies the variants *H1047R* and *E545K*, which yields queries $q_2 = \{PIK3CA, H1047R\}$ and $q_3 = \{PIK3CA, E545K\}$ that lead to D_2 and D_3. As soon as studies are found that contain the entities in a biomarker relationship, the entities and the studies are entered into the knowledge base. This process is then repeated until, theoretically, all published literature regarding the gene *PIK3CA* has been screened.

Our goal is to reduce the amount of documents which domain experts need to examine. To achieve this, an information retrieval system should rank documents high that are relevant to the query and should facilitate the identification of relevant entities to refine the query.

Classic approaches for document ranking, like tf-idf (Luhn, 1957); (Spärck Jones, 1972), or bm25 (Robertson and Zaragoza, 2009), and, for example, the Relevance Model (Lavrenko and Croft, 2001) for query refinement are established techniques in this setting. They are known to be robust and do not require data for training. However, as they are based on a bag-of-words model (BOW),

[1]https://www.ncbi.nlm.nih.gov/pubmed

they cannot represent a semantic relationship of entities in a document. This, for example, yields search results with highly ranked review articles that only *list* query terms, without the desired relationship between them. Therefore, we investigate approaches that model the semantic relationships between biomarker entities. This can either be addressed by combining BOW with rule-based filtering, or by supervised learning, i.e. learning-to-rank (LTR).

Our goal is, to tailor document ranking and query refinement to the task of the curator. This means that a document ranking model should assign a high rank to a document that contains the query entities in a biomarker relationship. A query refinement model should suggest additional query terms, i.e. biomarker entities, to the curator that are relevant to the current query. Given the complexity of entity relationships and the high variety of textual realizations this requires either effective feature engineering, or large amounts of training data for a supervised approach. The in-house data set of Molecular Health consists of 5833 labeled biomarker-facts, and 24 million unlabeled text documents from PubMed. Therefore, a good solution is to exploit the large amount of unlabeled data in a semi-supervised approach. Li et al. (2015) have shown that a neural auto-encoder with LSTMs (Hochreiter and Schmidhuber, 1997) can encode the syntactics and semantics of a text in a dense vector representation. We show that this representation can be effectively used as a feature for semi-supervised learning-to-rank and query refinement.

In this paper, we describe a feature engineering approach and a semi-supervised approach. In our experiments we show that the two approaches are, in comparison, almost on par in terms of performance and even improve in a joint model. In Section 2 we describe the neural auto-encoder, and then proceed in Section 3 to describe our models for document ranking and in Section 4 the models for query refinement.

2 Neural Auto-Encoder

In this study, we use an unsupervised method to encode text into a dense vector representation. Our goal is to investigate if we can use this representation as an encoding of the relations between biomarker entities.

Following Sutskever et al. (2014) Cho et al. (2014), Dai and Le (2015), and Li et al. (2015) we implemented a text auto-encoder with a Sequence-to-Sequence approach. In this model an encoder Enc produces a vector representation $\mathbf{v} = Enc(d)$ of an input document $d = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]$, with $\mathbf{w}_i$ being word embedding representations (Mikolov et al., 2013). This dense representation $\mathbf{v}$ is then fed to a decoder Dec, that tries to reconstruct the original input, i.e. $\hat{d} = Dec(\mathbf{v})$. During training we minimize $error(\hat{d}, d)$. After training we only use the $Enc(d)$ to encode the text. We want to explore if we can use Enc to encode the documents *and* the query. We will use the output of the document encoder Enc as features for a document ranking model and for a query refinement model.

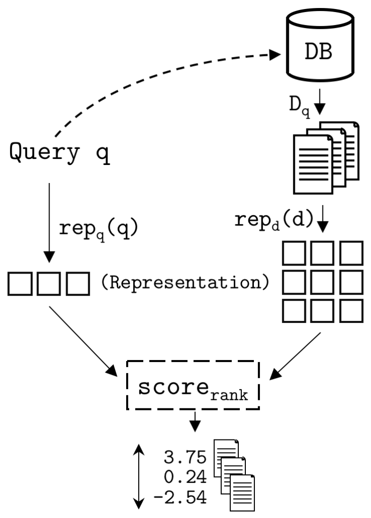

Figure 1: Document Ranking

3 Document Ranking

Information retrieval systems rank documents in the order that is estimated to be most useful to a user query by assigning a numeric score to each document. Our pipeline for document ranking is depicted in Figure 1: Given a query q, we first retrieve a set of documents D_q that contain all of the query terms. Then, we compute a representation $rep_q(q)$ for the query q, and a representation $rep_d(d)$ for each document $d \in D_q$. Finally, we compute the score with a ranker model $score_{rank}$.

For rep_d we need to find a representation for an arbitrary number of entity-type combinations, because a fact can consist of e.g. 3 *Genes*, 4 *Variants*, 1 *Drug*, 0 *Diseases* and 0 *Outcomes*. In the following, we describe several of the settings for $rep_q(q)$, $rep_d(d)$ and the ranker model.

3.1 Bag-of-Words Models

We have implemented two commonly used BOW models tf-idf and bm25. For these models the text representations $rep_q(q)$ and $rep_d(d)$ is the vector space model.

3.2 Learning-to-Rank Models

For the learning-to-rank models, we chose a multilayer perceptron (MLP) as the scoring function $score_{rank}$. In the following we explain how $rep_q(q)$ and $rep_d(d)$ are computed.

Feature Engineering Model We created a set of basic features: encoding the frequency of entity types, distance features between entity types, and context words of entities. In this model, features are query dependent and are computed on-demand by a feature function $f(q, d)$.

The algorithm to compute the distance feature is as follows: Given query q with entities $e \in q$ and document $d = [w_1, w_2, \ldots, w_n]$, with w being words in the document. Let $type(e)$ be the function that yields the entity type, f.ex. $type(e) = Gene$. Then, if $e_i, e_j \in q$ and there exists a $w_k = e_i, w_l = e_j$ then we add $|l - k|$ to the bucket of $\{type(e_i), type(e_j)\}$. To summarize the collected distances we compute $min()$, $max()$, $mean()$ and $std()$ over all collected distances for each bucket separately.

For the context words feature, we collected in a prior step a list of the top 20 words for each $\{type(e_i), type(e_j)\}$ bucket, i.e. we collect words that are between $w_k = e_i$ and $w_l = e_j$ if $|k - l| < 10$. We remove stop words, special characters and numbers from this list and also manually remove words using domain knowledge. The top 20 of remaining words for each $\{type(e_i), type(e_j)\}$ bucket are used as boolean indicator features.

Auto-Encoder Model In this model we use the auto-encoder from Section 2 to encode the query and the document. The input to the score function is the element-wise product, denoted by $\odot$, of the query encoding $rep_q(q) = Enc(q) = \mathbf{q}$ and the document encoding $rep_d(d) = Enc(d) = \mathbf{d}$:

$$score_{rank}(\mathbf{d}, \mathbf{q}) = MLP(\mathbf{d} \odot \mathbf{q}) \qquad (1)$$

To encode the queries we compose a pseudo text using the query terms. The input to the auto-encoder Enc are the word embeddings of the pseudo text for $rep_q(q)$ and the word embeddings of the document terms for $rep_d(d)$.

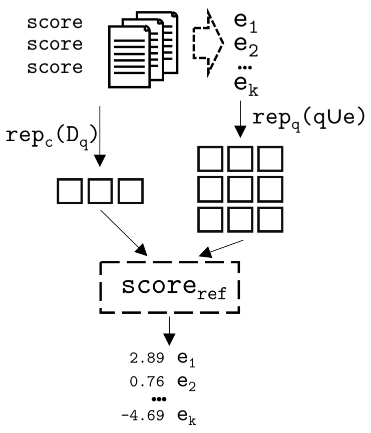

Figure 2: Query Refinement

4 Query Refinement

Query refinement finds additional terms for the initial query q that better describe the information need of the user (Nallapati and Shah, 2006). In our approach we follow Cao et al. (2008), in which the ranked documents D_q are used as pseudo relevance feedback. Our goal is to suggest relevant entities e that are contained in D_q and that are in a biomarker relationship to q. Therefore, we define a scoring function $score_{ref}$ for ranking of candidate entities e to the query q with respect to the retrieved document set D_q. See Figure 2 for a sketch of the query refinement pipeline. In the following, we describe several of the scoring functions.

4.1 Bag-of-Words Models

We implemented the two classic query refinement models the Rocchio algorithm (Rocchio and Salton, 1965) and Relevance Model.

4.2 Auto-Encoder Model

In this model, we also try to exploit the auto-encoding of the query. The idea is as follows: (i) Given a list of documents and their scores $D_q = [(\mathbf{d}_1, s_1), (\mathbf{d}_2, s_2), \ldots, (\mathbf{d}_n, s_n)]$ for a query q from the previous step, we use the ranking score s_i as pseudo-relevance feedback to create a pseudo document $rep_c(\mathbf{D}, \hat{\mathbf{s}}) = \sum_i^n \mathbf{d}_i \hat{s}_i = \hat{\mathbf{d}}$. The scores s are normalized so that they are non-negative and $\sum_i \hat{s}_i = 1$, see Appendix A.1. (ii) From all entities $e_i \in D_q \backslash q$ we generate new query encoding $rep_q(q \cup e_i) = Enc(q \cup e_i) = \hat{\mathbf{q}}_{e_i}$ (iii) We rank the entities based on the pseudo document using the scoring function

$$score_{ref}(\hat{\mathbf{q}}_{e_i}, \hat{\mathbf{d}}) = MLP(\hat{\mathbf{d}} \odot \hat{\mathbf{q}}_{e_i}) \qquad (2)$$

i.e. we propose those entities as a query refinement that agree with the most relevant documents.

5 Experiments

In this section, we first explain our evaluation strategy to assess the performance of the respective models for document ranking and query refinement. Subsequently, we describe the settings for the data and the results of the experiments that we have conducted.

5.1 Evaluation Protocol

Document ranking models are evaluated by their ability to rank relevant documents higher than irrelevant documents. Query refinement models are evaluated both, by their ability to rank relevant query terms high, and by the recall of retrieved relevant documents when the query automatically is refined by the 1st, 2nd and 3rd proposed query term. We evaluate our models using mean average precision (MAP) (Manning et al., 2008, Chapter 11) and normalized discounted cumulative gain (nDCG) (Järvelin and Kekäläinen, 2000).

For both the document ranking and query refinement approach we interpret a biomarker-fact as a perfect query and the corresponding papers as the true-positive (or relevant) papers associated with this query. In this way, we use the curated facts as document level annotation for our approach. Because we want to assist the iterative approach of curators in which they refine an initially broad query to ever narrower searches, we need to create valid partial queries and associated relevant documents to mimic this procedure. Therefore, we generate sub-queries, which are partials of the facts. We generated two data sets: one for document ranking and one for query refinement. For document ranking, we generated all distinct subsets of the facts. For query refinement, we defined the eliminated entities (of the sub-query generation process) as true-positive refinement terms. For both data sets, we use all associated documents, of the original biomarker-fact, as true-positive relevant documents.

5.2 Data

Unlabeled Data As unlabeled data we use ∼24 Million abstracts of PubMed. To automatically annotate PubMed abstracts with disambiguated biomarker entities, we use a tool set that has been developed together with biomedical curators. It employs "ProMiner"[2] (Hanisch et al., 2005) for *Protein/Gene* and *Disease* entity recognition and regular expression based text scanning using synonyms from "DrugBank"[3] and "PubChem"[4] for the identification of *Drugs* and manually edited regular expressions, relating to "HGVS"[5] standards, to retrieve *Variants*. We restricted the PubMed documents to include at least one entity of type *Gene*, *Drug* and *Disease* leaving us with 2.7 Million documents. Additionally we replaced the text of every disambiguated entity with its id.

Labeled Data As labeled data we use a knowledge base that contains a set of 5833 hand curated {*Gene(s) - Variant(s) - Drug(s) - Disease(s) - Outcome*} biomarker-facts and their associated papers that domain experts extracted from ∼1200 full-text documents. We only keep facts in which the disambiguated entities are fully represented in the available abstracts. This restricted our data set to a set of 1181 distinct facts. The 4 top curated genes are *EGFR* (29%), *BRAF* (13%), *KRAS* (8%), and *PIK3CA* (5%).

Cross Validation To exploit all of our labeled data for training and testing we do 4-fold cross-validation. Because in our scenario a curator starts with an initial entity of type *Gene* we have generated our validation and test sets based on *Gene*s, instead of randomly sampling facts. This also guarantees us to never have the same sub-query included in the training, validation and test set. In total we have built 12 different splits of our data set basing the validation and test set each on a different gene. The respective training sets are built with all remaining facts that do not include the validation and test gene. Statistics can be found in Table 1.

5.3 Training of Embeddings and Auto-Encoder

The training data for the embeddings and the auto-encoder are the PubMed abstracts described in the previous Section 5.2. We trained the embeddings with Skip-Gram. For the vocabulary, we keep the top 100k most frequent words, while making sure all known entities are included. We use a window size of 10 and train the embeddings for 17

[2]https://www.scai.fraunhofer.de/en/business-research-areas/bioinformatics/products/prominer.html

[3]https://www.drugbank.ca

[4]https://pubchem.ncbi.nlm.nih.gov

[5]http://www.hgvs.org

Validation		**Testing**		**Train**
Gene	#Data	Gene	#Data	#Data
BRAF	1135	EGFR	1822	4386
BRAF	1075	KRAS	968	6310
BRAF	1088	PIK3CA	549	6944
EGFR	1700	BRAF	1241	4386
EGFR	1475	KRAS	968	5536
EGFR	1605	PIK3CA	549	5957
KRAS	573	EGFR	1822	5536
KRAS	804	BRAF	1241	6310
KRAS	778	PIK3CA	549	7754
PIK3CA	298	EGFR	1822	5957
PIK3CA	382	BRAF	1241	6944
PIK3CA	367	KRAS	968	7754

Table 1: Statistics about the train/validation/test splits

epochs. We normalize digits to ”0” and lowercase all words. Tokenization is done by splitting on white space and before and after special characters.

For both, the encoder and the decoder, we use two LSTM cells per block with hidden size 400 each. We skip unknown tokens and feed the words in reverse order for the decoder following Sutskever et al. (2014). The auto-encoder was trained for 15 epochs using early stopping.

5.4 Document Ranking

In this section, we describe the document ranking models, their training and then discuss the results of their evaluation.

Models We evaluate the BOW models (*tf-idf, bm25*) (Section 3.1) and the two LTR models using feature engineering (*feat*) and the auto-encoded features (*auto-rank*) (Section 3.2). We also evaluate an additional set of models to investigate if maybe simpler solutions can be competitive. See Table 2 for an overview over all ranking models.

(a.) A simpler solution than learning a MLP for a score function is to compute the similarity between $\mathbf{q} = Enc(q)$ and the document encoding $\mathbf{d} = Enc(d)$. Therefore, we use the cosine similarity as scoring function between the vector representations $\mathbf{q}$ and $\mathbf{d}$ (*auto/cosine*).

(b.) Instead of encoding the documents and queries with the auto-encoder, we encode the documents and queries based on their tf-idf weighted embeddings, i.e. $\mathbf{q} = \sum tfidf(q_i) * \mathbf{w}_{q_i}$. Similarly to the auto-rank model, the input to the classifier MLP is the element-wise product of the query encoding and the document encoding (*emb*).

(c.) Due to promising results of the *auto-rank* model, *bm25*, and the *feat* model, we also tested combinations of them. We tested the concatenation of the the *bm25* score with the *auto-rank* features (*auto-rank* + *bm25*) as well as the concatenation of *feat* with the *auto-rank* features (*auto-rank* + *feat*).

Training We train our models with Adam (Kingma and Ba, 2014) and tune the initial learning rate, the other parameters are kept default of TensorFlow[6]. We use a pairwise hinge loss (Chen et al., 2009) and compare relevant documents with irrelevant documents.

The ranking score function is parameterized by a MLP for which the number of layers is a hyperparameter which is tuned using grid-search. The input layer size is based on the number of input features. To limit the total number of parameters, we decrease the layer size while going deeper, i.e. layer i has size $l_i = \frac{b-i+1}{b}|u|$, with b being the depth of the network, $|u|$ the number of input features. For activation functions between layers we use ReLU (Glorot et al., 2011).

We employ grid search over the hyperparameters: dropout: $[0.3, 0.2, 0.1, 0.0]$, number of layers: $[1, 2, 3, 4]$, learning rates for Adam: $[0.0005, 0.001]$, batch size: $[40, 60]$, max 400 epochs. We conducted hyper-parameter tuning for each model and validation/test split separately. The best parameters for the models using the auto-encoded features were: 1 layer, dropout $p \in [0.3, 0.2]$ with a batch size of 60 and learning rate 0.0005. The feat models were best with 1-2 layers, dropout 0.0 and a learning rate at 0.001.

The BOW models as well as *auto/cosine* were only computed for the respective validation and test sets.

Results In Table 3 we have listed the average MAP and nDCG scores of the test sets. The *tf-idf* model is outperformed by most of the other models. However, *bm25*, which additionally takes the length of a document into account, performs very well. *tf-idf* and *bm25* have the major benefit of fast computation.

The *feat* model slightly outperforms the *auto-*

[6]Tensorflow V 1.3 https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer

Model	$rep_q(q)$	$rep_d(d)$	**score**
tf-idf	BOW	BOW	dot product
bm25	BOW	BOW + doc length	bm25
emb	**q** = tf-idf BOW ·**w**	**q** = tf-idf BOW ·**w**	$MLP(\mathbf{q} \odot \mathbf{d})$
feat	$f(q,d)$		$MLP(f(q,d))$
auto/cosine	$\mathbf{q} = Enc(q)$	$\mathbf{d} = Enc(d)$	$cos(Enc(q), Enc(d))$
auto	$\mathbf{q} = Enc(q)$	$\mathbf{d} = Enc(d)$	$MLP(\mathbf{q} \odot \mathbf{d})$
auto + bm25	$\mathbf{q} = Enc(q)$, BOW	$\mathbf{d} = Enc(d)$, BOW + doc length	$MLP(\mathbf{q} \odot \mathbf{d}$, bm25)
auto + feat	$\mathbf{q} = Enc(q)$	$f(q,d)$ $\mathbf{d} = Enc(d)$	$MLP(\mathbf{q} \odot \mathbf{d}, f(q,d))$

Table 2: Query and document representation for ranking models

Test	**Metric**	**tf-idf**	**bm25**	**emb**	**feat**	**auto/ cosine**	**auto-rank**	**auto-rank + bm25**	**auto-rank + feat**
EGFR	MAP	0.289	0.632	0.310	0.575	0.054	0.545	0.588	**0.699**
	nDCG	0.424	0.728	0.460	0.695	0.129	0.653	0.716	**0.810**
KRAS	MAP	0.327	0.610	0.466	0.609	0.058	0.575	0.774	**0.820**
	nDCG	0.456	0.723	0.592	0.712	0.145	0.688	0.867	**0.914**
BRAF	MAP	0.342	0.656	0.427	0.704	0.063	0.563	0.702	**0.812**
	nDCG	0.480	0.751	0.572	0.802	0.163.	0.671	0.820	**0.901**
PIK3CA	MAP	0.341	0.633	0.486	0.625	0.079	0.541	0.779	**0.810**
	nDCG	0.473	0.729	0.617	0.718	0.171	0.656	0.859	**0.895**

Table 3: Test Scores Document Ranking

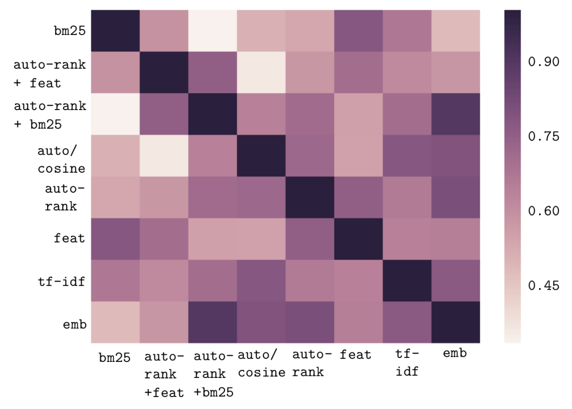

Figure 3: Correlation of Document Ranking Models

rank model. The distance features are a strong indicator for the semantic dependency between entities. These relationships need to be learned in the *auto-rank* model.

The cosine similarity of a query and a document (*auto/cos*) does not yield a good result. This shows that the auto-encoder has learned many features, most of which do not correlate with our task. We also find that *emb* does not yield an equal performance to *auto-rank*. The combination of the *auto-rank + feat* model is slightly better than the *auto-rank + bm25* model, both of which have the overall best performance. This shows, that the auto-encoder learns something orthogonal to term frequency and document length. The best model with respect to document ranking is the *auto-rank + feat* model.

In Figure 3 we show the correlation between the different models. Interestingly, the *bm25* and the *feat* strongly correlate. However, the scores of *bm25* do not correlate with the scores of the combination of *auto-rank* and *bm25*. This indicates, that the model does not primarily learn to use the *bm25* score but also focuses on the the auto-encoded representation. This underlines the hypothesis that the auto-encoder is able to represent latent features of the relationship of the query terms in the document.

Influence of the Data It is interesting to see that the learned models do not perform well for the *EGFR* set. The reason for this might be that testing on it reduces the amount of training data substantially, as *EGFR* is the best curated gene and thus the largest split of the data set.

In a manual error analysis we compared the rankings of four of the best models (*bm25*, *feat*, *auto-rank*, and *auto-rank* + *bm25*). We observe cases where the *auto-rank* model is unable to detect, when similar entities are used, i.e. entities like *Neoplasm* and *Colorectal Neoplasm*. In these cases the *bm25* helps, as it treats different words as different features. However, both *bm25* and the *feat* models rank reviews high, that only *list* query terms. For example, when executing the query {*BRAF, PIK3CA, H1047R*}, these models rank survey articles high (i.e. (Li et al., 2016); (Janku et al., 2012); (Chen et al., 2014)).

The *auto-rank* model on the other hand ranks those document high, for which each entities are listed in a semantic relationship (i.e. (Falchook et al., 2013)).

5.5 Query Refinement

In this section we describe our training approaches for query refinement and discuss our findings. The pseudo relevance feedback for the query refinement is based on the ranked documents from the previous query. For our experiments we chose the second best document ranker (*auto-rank* + *bm25*) from the previous experiments, because our prototype implementation for *auto-rank* + *feat* was computationally too expensive.

Models We combined both the auto-encoder features with the candidate terms of the respective BOW models (*auto-ref* + *rocch* + *relev*). In order to identify if the good results of this combination are due to the BOW models, or if the auto-encoded features have an effect, we trained a MLP with the same amount of parameters, but only use the features of the two BOW models as input (*rocchio* + *relev*).

Training For training, we use the same settings for query refinement as for document ranking and again use a pairwise hinge loss. Here we compare entities that occur in the facts with randomly sampled entities which occur in the retrieved documents.

Due to limitations in time we were only able to test our query refinement models on one validation/test split. We chose to use the split data set of genes *KRAS* and *PIK3CA* for validation and testing respectively. We have restricted our models to only regard the top 50 ranked documents for refinement.

Results To evaluate the ranking of entity terms we have computed nDCG@10, nDCG@100 and MAP, see Table 4 for the results. We also compute Recall@k of relevant documents for automatically refined queries using the 1st, 2nd and 3rd ranked entities. The scores can be found in Table 5.

Tables 4 and 5 show that the Relevance Model outperforms the Rocchio algorithm in every aspect. Both models outperform the auto-encoder approach (*auto-ref*). We suspect that summing over the encodings distorts the individual features too much for a correct extraction of relevant entities to be possible.

The combination of all three models (*auto-ref* + *rocchio* + *relevance*) outperforms the other models in most cases. Especially the performance for ranking of entity terms is increased using the auto-encoded features. However, it is interesting to see that the *rocchio* + *relevance* model outperforms the recall for second and third best terms. This indicates that for user-evaluated term suggestions, the inclusion of the auto-encoded features is advisable. For automatic query refinement however, in average, this is not the case.

Variants	**Diseases**	**Drugs**
H1047R	Color. Neop.	Lapatinib
~~V600E~~	Liposarcoma	Mitomycin
~~T790M~~	Adenocarcin.	Linsitinib
E545K	Glioblastoma	Dactolisib
E542K	Stomach Neop.	Pictrelisib

Table 6: Refinement Terms for Query {PIK3CA}

Query Refinement Example In Table 6 we show the top ranked entities of type *Variants*, *Diseases* and *Drugs* for the query {*PIK3CA*}. While the diseases and the drugs are all relevant, *V600E* and *T790M* are in fact not variants of the gene *PIK3CA*.

However, when refining the query {*PIK3CA*, *V600E*, *BRAF*, *H1047R*, *Dabrafenib*}, the top ranked diseases are [*Melanoma, Neoplasms, Carcinoma Non Small Cell Lung (CNSCL), Thyroid Neoplasms, Colorectal Neoplasms*]. Using *Melanoma* for refinement, retrieves the top ranked paper (Falchook et al., 2013) which perfectly includes all these entities in a biomarker relationship.

Metrics	rocchio	relevance model	auto-ref	rocchio + relevance	auto-ref + rocchio + relevance
nDCG@10	0.232	0.274	0.195	0.341	**0.464**
nDCG@100	0.360	0.397	0.329	0.439	**0.536**
MAP	0.182	0.223	0.156	0.270	**0.386**

Table 4: Ranked Entity Scores for KRAS Validation and PIK3CA Testing

Metrics	Top n-th Entity	rocchio	relevance model	auto-ref	rocchio + relevance	auto-ref + rocchio + relevance
Recall@10	1	0.594	0.603	0.272	0.574	**0.696**
	2	0.535	0.561	0.339	**0.580**	0.522
	3	0.533	0.544	0.366	**0.555**	0.458
Recall@40	1	0.683	0.691	0.307	0.680	**0.779**
	2	0.603	0.633	0.374	**0.649**	0.586
	3	0.610	0.623	0.402	**0.626**	0.522

Table 5: Refinement Recall Scores for KRAS Validation and PIK3CA Testing

6 Related Work

The focus of research in this domain has primarily targeted the extraction of entity relations. Peng and Lu (2017), Peng et al. (2015), and Li et al. (2017) try to extract *Protein-Protein* relationships. Asada et al. (2017) try to extract *Drug-Drug* interaction and Lee et al. (2018) target the extraction of *Mutation-Gene* and *Mutation-Drug* relations. Jameson (2017) have derived a document ranking approach for PubMed documents using word embeddings trained on all PubMed documents. Xu et al. (2017) propose using auto-encoders on the vector-space model in a supervised setting for information retrieval and show that it improves performance. The quality of biomedical word embeddings was investigated by Th et al. (2015) and Chiu et al. (2016). Dogan et al. (2017) have developed an open source data set to which we would like to adapt our approach. Sheikhshab et al. (2016) have developed a novel approach for tagging genes which we would like to explore.

Palangi et al. (2016) use LSTMs to encode the query and document and use the cosine similarity together with the click-through data as features for ranking in a supervised approach. Cao et al. (2008) define a distance based feature-engineered supervised learning approach to identify good expansion terms. They try to elaborate if the selected terms for expansion are useful for information retrieval by identifying if the terms are actually related to the initial query. Nogueira and Cho (2017) have introduced a reinforcement learning approach for query refinement using logging data. They learn a representation of the text and the query using RNNs and CNNs and reinforce the end result based on recall of a recurrently expanded query. Sordoni et al. (2015) have developed a query reformulation model based on sequences of user queries. They have used a Sequence-to-Sequence model using RNNs to encode and decode queries of a user.

7 Conclusion

We have considered several approaches for document ranking and query refinement by investigating classic models, feature engineering and, due to the large amount of unlabeled data, a semi-supervised approach using a neural auto-encoder.

Leveraging the large amounts of unlabeled data to learn an auto-encoder on text documents yields semantically descriptive features that make subsequent document ranking and query refinement feasible. The combination with BOW features increases the performance substantially, which for our experiments, outputs the best results, for both document ranking and query refinement.

We were able to achieve promising results, however, there is a wide range of Sequence-to-Sequence architectures and text encoding strategies, therefore, we expect that there is room for improvement.

References

Masaki Asada, Makoto Miwa, and Yutaka Sasaki. 2017. Extracting drug-drug interactions with attention cnns. In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 9–18.

Francis Charles Brunicardi, Richard A. Gibbs, David A. Wheeler, John Nemunaitis, William Fisher, John Goss, and Changyi Johnny Chen. 2011. Overview of the development of personalized genomic medicine and surgery. *World Journal of Surgery*, 35:1693–1699.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 243–250.

Jing Chen, Fang Guo, Xin Shi, Lihua Zhang, Aifeng Zhang, Hui Jin, and Youji He. 2014. BRAF V600E mutation and KRAS codon 13 mutations predict poor survival in Chinese colorectal cancer patients. *BMC Cancer*, 14(1):802.

Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 315–323.

Billy Chiu, Gamal K. O. Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing, BioNLP@ACL 2016, Berlin, Germany, August 12, 2016*, pages 166–174.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3079–3087.

Rezarta Islamaj Dogan, Andrew Chatr-aryamontri, Sun Kim, Chih-Hsuan Wei, Yifan Peng, Donald C. Comeau, and Zhiyong Lu. 2017. Biocreative VI precision medicine track: creating a training corpus for mining protein-protein interactions affected by mutations. In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 171–175.

Gerald S. Falchook, Jonathan C. Trent, Michael C. Heinrich, Carol Beadling, Janice Patterson, Christel C. Bastida, Samuel C. Blackman, and Razelle Kurzrock. 2013. BRAF Mutant Gastrointestinal Stromal Tumor: First report of regression with BRAF inhibitor dabrafenib (GSK2118436) and whole exomic sequencing for analysis of acquired resistance. *Oncotarget*, 4(2).

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 315–323.

Daniel Hanisch, Katrin Fundel, Heinz-Theodor Mevissen, Ralf Zimmer, and Juliane Fluck. 2005. Prominer: rule-based protein and gene entity recognition. *BMC Bioinformatics*, 6(S-1).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Anthony Jameson. 2017. A tool that supports the psychologically based design of health-related interventions. In *Proceedings of the 2nd International Workshop on Health Recommender Systems co-located with the 11th International Conference on Recommender Systems (RecSys 2017), Como, Italy, August 31, 2017.*, pages 39–42.

Filip Janku, Jennifer J. Wheler, Aung Naing, Vanda M. T. Stepanek, Gerald S. Falchook, Siqing Fu, Ignacio Garrido-Laguna, Apostolia M. Tsimberidou, Sarina A. Piha-Paul, Stacy L. Moulder, J. Jack Lee, Rajyalakshmi Luthra, David S. Hong, and Razelle Kurzrock. 2012. PIK3CA Mutations in Advanced Cancers: Characteristics and Outcomes. *Oncotarget*, 3(12).

Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *SIGIR 2000: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece*, pages 41–48.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 120–127.

Kyubum Lee, Byounggun Kim, Yonghwa Choi, Sunkyu Kim, Won-Ho Shin, Sunwon Lee, Sungjoon Park, Seongsoon Kim, Aik Choon Tan, and Jaewoo Kang. 2018. Deep learning of mutation-gene-drug relations from the literature. *BMC Bioinformatics*, 19(1):21:1–21:13.

Gang Li, Cathy H. Wu, and K. Vijay-Shanker. 2017. Noise reduction methods for distantly supervised biomedical relation extraction. In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 184–193.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents.

Wan-Ming Li, Ting-Ting Hu, Lin-Lin Zhou, Yi-Ming Feng, Yun-Yi Wang, and Jin Fang. 2016. Highly sensitive detection of the PIK3CA H1047R mutation in colorectal cancer using a novel PCR-RFLP method. *BMC Cancer*, 16(1):454.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Teri A. Manolio. 2010. Genomewide Association Studies and Assessment of the Risk of Disease. *New England Journal of Medicine*, 363(2):166–176.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Ramesh Nallapati and Chirag Shah. 2006. Evaluating the quality of query refinement suggestions in information retrieval.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 574–583.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab K. Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(4):694–707.

Yifan Peng, Samir Gupta, Cathy H. Wu, and K. Vijay-Shanker. 2015. An extended dependency graph for relation extraction in biomedical texts. In *Proceedings of the Workshop on Biomedical Natural Language Processing, BioNLP@IJCNLP 2015, Beijing, China, July 30, 2015*, pages 21–30.

Yifan Peng and Zhiyong Lu. 2017. Deep learning for extracting protein-protein interactions from biomedical literature. In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 29–38.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Joseph Rocchio and Gerard Salton. 1965. Information search optimization and interactive retrieval techniques. In *AFIPS '65 (Fall, part I)*.

Golnar Sheikhshab, Elizabeth Starks, Aly Karsan, Anoop Sarkar, and Inanç Birol. 2016. Graph-based semi-supervised gene mention tagging. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing, BioNLP@ACL 2016, Berlin, Germany, August 12, 2016*, pages 27–35.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 553–562.

Karen Spärck Jones. 1972. A Statistical Interpretation of Term Specificity and its Retrieval. *Journal of Documentation*, 28(1):11–21.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Muneeb Th, Sunil Kumar Sahu, and Ashish Anand. 2015. Evaluating distributed word representations for capturing semantics of biomedical concepts. In *Proceedings of the Workshop on Biomedical Natural Language Processing, BioNLP@IJCNLP 2015, Beijing, China, July 30, 2015*, pages 158–163.

Bo Xu, Hongfei Lin, Yuan Lin, and Kan Xu. 2017. Learning to rank with query-level semi-supervised autoencoders. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2395–2398.

A Supplemental Material

A.1 Normalizing Document Scores for Query Refinement

Because we used a hinge loss instead of cross entropy loss in the ranking model, we cannot interpret the scores **s** of the ranker as logits. While we do not know the magnitude of the ranker score, we do, however, expect the scores to be positive for relevant documents. If however many documents have scores below zero, this should also be regarded. Based on this, we have defined a normalization setting of the document scores:

$$s_{min} = \text{min}(\text{min}(\mathbf{s}), 0.0) \tag{3}$$

$$\hat{\mathbf{s}} = \frac{\mathbf{s} - s_{min}}{\sum_i^{|\mathbf{S}|}(s_i - s_{min})} \tag{4}$$