Background Knowledge Injection for Interpretable Sequence Classification

Severin Gsponer[1], Luca Costabello[2], Chan Le Van[2], Sumit Pai[2], Christophe Gueret[2], Georgiana Ifrim[1], and Freddy Lecue[3]

[1] Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland, {firstname.lastname}@insight-centre.org
[2] Accenture Labs, Dublin, Ireland, {firstname.lastname}@accenture.com
[3] Inria, Sophia Antipolis, France, freddy.lecue@inria.fr

Abstract. Sequence classification is the supervised learning task of building models that predict class labels of unseen sequences of symbols. Although accuracy is paramount, in certain scenarios interpretability is a must. Unfortunately, such trade-off is often hard to achieve since we lack human-independent interpretability metrics. We introduce a novel sequence learning algorithm, that combines (i) linear classifiers - which are known to strike a good balance between predictive power and interpretability, and (ii) background knowledge embeddings. We extend the classic subsequence feature space with groups of symbols which are generated by background knowledge injected via word or graph embeddings, and use this new feature space to learn a linear classifier. We also present a new measure to evaluate the interpretability of a set of symbolic features based on the symbol embeddings. Experiments on human activity recognition from wearables and amino acid sequence classification show that our classification approach preserves predictive power, while delivering more interpretable models.

1 Introduction

Sequence classification - the task of assigning classes to sequences of atomic symbols - occurs in a multitude of applicative scenarios such as ubiquitous computing, bioinformatics, finance, and security surveillance [22]. A concrete example is the determination of protein types base solely on the amino-acid sequence. Deep neural architectures deliver excellent predictive power, at the expense of human interpretability [3] and high demand on computational resources. Other sequence classification approaches provide better interpretability (e.g. linear models), but this is often achieved at the expense of predictive power. Not only is such an accuracy-interpretability trade-off hard to achieve, but comparing models interpretability is often left to manual inspection, as there are no agreed-upon, human-independent measures available. At the same time, a large number of textual and graph-based background knowledge bases are available on the web. Nevertheless, there are no existing works that merge predictive models trained on sequences of symbols with prior knowledge from external knowledge bases.

In this work, we focus on the problem of designing an interpretable model for sequence classification. The problem consists of two parts: i) conceiving the model itself, and ii) validating the improvement in interpretability with a proper

metric. Unlike existing sequence classification models, the intuition behind our work is that auxiliary, external background information can i) enhance the interpretability of sequence classification models, and ii) help measure such interpretability. We show that linear models for classification - which are known to strike a good balance between predictive power and interpretability - can be enriched with auxiliary background knowledge to obtain a quantifiable improvement in the interpretability of their features, without affecting the predictive power. Our contribution (Figure 1) includes:

- **Emb-SEQL**: a feature selection and learning algorithm for sequence classification that uses external embeddings to refine the selection of candidate features.
- **Semantic Fidelity**: a metric to quantify the interpretability of features extracted from symbolic sequences. The metric casts the problem into computing distances in a background knowledge embedding space, and does not depend on human-grounded evaluation protocols.

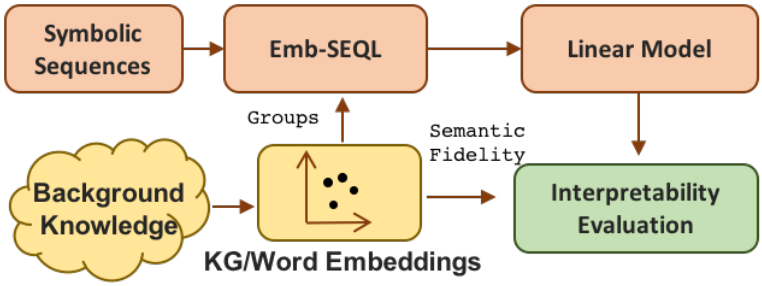

Fig. 1. Simplified overview of our contribution: Injecting domain knowledge for interpretable sequence classification.

We evaluate our approach on human activity recognition from wearables (HAR) and amino acid sequence classification. We assess both predictive power and interpretability, experimenting with pre-trained word embeddings and knowledge graph embeddings. We find that using auxiliary knowledge to refine the selection of candidate features results in more interpretable models. At the same time, it does not reduce the predictive power of the learned model. Links to data and code will be inserted in the final version of the paper.

2 Related Work

Sequence Classification: Learning classification models for symbolic sequences often uses the presence or the frequency of consecutive groups of k symbols, so called k-mers (or n-grams in text processing) as features [22]. Support Vector Machines (SVMs) show promising results: specific string kernels have been proposed, as well as implementation tricks that improve their efficiency [7,18,19]. Markov Models and Hidden Markov Models [15] model the probability distribution of sequences for each class separately and assign the class with the highest likelihood to unseen sequences at inference time. Current state-of-the-art results

are held by Convolutional Neural Networks (CNN) that operate on sequences of characters, which have been successfully applied to sequences [24,1]. Nevertheless, CNNs and SVMs are black box models and have poor interpretability.
Background Knowledge Injection: Background knowledge is typically used to improve accuracy: auxiliary knowledge can be encoded as rules, for more accurate relation extraction [16], or to predict missing links in knowledge graphs [11]. There have been attempts to incorporate semantic monotonic constraints derived from background knowledge [4], but not for sequence classification.
Interpretability Metrics: [3] discuss evaluation protocols to assess the interpretability of machine learning models. They take into account human-grounded experiments - with real-world and simplified tasks. Besides, they also acknowledge the need for *functionally-grounded* protocols that replace human intervention with proxy tasks. A simple proxy to compare linear classifiers is measuring the size of the model (number of features with non zero weights) - the assumption being that the smaller the model, the higher the interpretability. Nevertheless, this is an over-simplistic assumption, as size does not capture the semantics of the model features [4].

3 Preliminaries

3.1 Sequence Classification

Learning a mapping from sequences of symbols to categorical labels is commonly known as sequence classification. Let $D = \{(s_1, y_1), (s_2, y_2), \ldots, (s_N, y_N)\}$ be a sequence database of instance-label pairs, where s_i is a sequence and $y_i \in L$ the corresponding label. The goal of sequence classification is to learn a mapping ξ from the sequence database D so that we can predict the label of a yet unlabeled sequence $s \in S$. Formally such a mapping is a function $\xi : S \to L$ where S is the set of all possible sequences and $L = \{c_1, c_2, \ldots, c_N\}$ the set of class labels. A sequence $s_i \in S$ has the following form $s_i = \langle \sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{in_i} \rangle$ and each of the individual symbols σ_{ij} belongs to a predefined finite alphabet Σ. For example, if $\Sigma = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ a sequence could be $s_1 = \langle \mathrm{B}, \mathrm{A}, \mathrm{B}, \mathrm{C} \rangle$. Note that the lengths n_i of sequences is variable.

A k-mer is a sequence of k consecutive symbols, e.g., $k' = \langle \mathrm{B}, \mathrm{A}, \mathrm{B} \rangle$. We write $k' \subseteq s_i$ and say k' is present in s_i if an exact match of k' is found in s_i. Given this definition and an enumeration schema of all k-mers present in the training data, we can represent a sequence s_i as a binary vector: $\mathbf{x}_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{in})^T, x_{ij} \in \{0, 1\}, i = 1, \ldots, N$, where $x_{ij} = 1$ means that k-mer k_j occurs in sequence s_i.

Such a representation allows us to learn a linear model, i.e., a parameter vector $\boldsymbol{\beta}$ of feature weights to predict the class label of a sequence s_i by setting $y_i = sgn(\boldsymbol{\beta}^T \mathbf{x}_i) = \xi(s_i)$. Although linear models are not powerful enough to capture non-linear relationships, by working in a very complex feature space (e.g., all k-mers) it is possible to learn powerful models, similar to the kernel trick applied by kernel Support Vector Machines [6].

3.2 SEQL

Although the entire *k-mer space* is huge and in practice infeasible to generate explicitly, it can still be used by exploiting the nested structure of the feature-

space using SEQL [6]. In this work we adopt SEQL a linear sequence classifier algorithm, as we want to learn a model that is interpretable but still achieves high accuracy. The main idea behind SEQL is to use a greedy coordinate gradient descent with the Gauss-Southwell rule [13] which allows to avoid the explicit generation of the feature vectors [5]. A key step of this approach is the efficient search for the current *best* k-mer, in the sense of maximum absolute gradient value, followed by an update of the corresponding weight value β. These two steps are executed iteratively until a convergence threshold is reached. The search part itself is realized with a branch-and-bound tree search which is made feasible by a bound on the gradient value of k-mers based on its own sub-k-mers. In particular, each iteration starts by computing the gradient values of all 1-mers whereby the best gradient value found so far is saved in τ. For each of the 1-mer k' the corresponding upper bound $\mu(k')$ is computed. The sub-tree starting at k' can be pruned whenever $\mu(k') \leq \tau$ otherwise we expand k' and repeat the procedure. This search procedure allows to find the *best* k-mer in an efficient and timely manner. The resulting model is a weighted list of k-mers which is easier to understand by humans. SEQL has support for two classification losses i) logistic loss and ii) squared hinge loss; here, we use i) to learn linear binary sequence classification models.

Moreover, SEQL goes beyond traditional k-mers, since it has the ability to use wildcards within the generated k-mers by using the $*$-character. Such wildcard allows k-mers with gaps, which leads to more general features. Nevertheless this is computationally expensive [6].

3.3 Word Embeddings

Word embeddings are representation learning techniques widely adopted in natural language processing. They map words in a text corpus to a low-dimensional, continuous vector space. Such vectors act as representations of terms in a n-dimensional metric space. Word embeddings are mostly generated by processing word co-occurrences, or by using neural architectures, the most popular models being word2vec [9] and GloVe [14]. Popular pre-trained word embeddings collections such as ConceptNet Numberbatch [20] and GloVe[4] are available on the web. The main shortcoming of word embeddings is that single vectors may represent words that carry multiple meanings.

3.4 Knowledge Graph Embedddings

Knowledge graphs are graph-based knowledge bases whose facts are modeled as relationships between entities. Examples are DBpedia, WordNet, and YAGO. Formally, a knowledge graph $\mathcal{G} = \{(sub, pred, obj)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples in the form $t = (sub, pred, obj)$, each including a subject $sub \in \mathcal{E}$, a predicate $pred \in \mathcal{R}$, and an object $obj \in \mathcal{E}$. $\mathcal{E}$ and $\mathcal{R}$ are the sets of all entities and relation types of $\mathcal{G}$.

Knowledge graph embedding models are neural architectures that encode concepts from a knowledge graph (i.e. entities $\mathcal{E}$ and relation types $\mathcal{R}$) into low-dimensional, continuous vectors $\in \mathbb{R}^k$. Such *knowledge graph embeddings* have

[4] https://nlp.stanford.edu/projects/glove/

many applications, e.g., in knowledge graph completion, entity resolution, and link-based clustering [12]. Knowledge graph embeddings are learned by training a neural architecture over a knowledge graph. Although such architectures vary, the training phase always consists in minimizing a loss function (usually negative log-likelihood or hinge loss) $\mathcal{L}$ that includes a *scoring function* $f_m(t)$, i.e., a model-specific function that assigns a score to a triple t. The optimization procedure learns optimal embeddings by minimizing $\mathcal{L}$, such that the model assigns high scores to true statements, and low scores to statements unlikely to be true.

4 Method

In this section we describe Emb-`SEQL`, our background knowledge-enriched sequence classification model. We also present the Semantic Fidelity, the metric that we use to assess the interpretability of the features learned by Emb-`SEQL`.

4.1 Emb-SEQL

One of the drawbacks of the k-mer-based approach of `SEQL` is that it fully relies on matching exact k-mers. The $*$-wildcard relaxes this constraint but it is very general as it allows an arbitrary symbol. As an alternative approach, we introduce the concept of *groups*. The main intuition behind these *groups* is that there exist symbols in the alphabet that are exchangeable in certain situations. Conceptually, *groups* form a new symbol that can be considered as `OR` combination of multiple base symbols from the original alphabet. We use these new symbols to extend the all-k-mer representation that `SEQL` uses and write them, similar to a regular expressions, as (A|B). *Groups* can be formed by hand but we are more interested in forming them automatically by exploiting background knowledge.

Symbols in more complex alphabets (e.g., Activities, NLP) often have relationships between each other in the sense that some symbols are semantically closer than others. We use the embedded representations of symbols to measure such closeness and form *groups* based on this measurement. To find sensible *groups* automatically we first map all base symbols into the embedding space followed by clustering. Various (overlapping) clustering techniques can be used for this task. We adopt a simple radius-based approach: for each symbol σ in the alphabet a *group* is formed by aggregating all the symbols that fall within a fixed radius $r \in \mathbb{R}$ around the embedding of the symbol σ. After collecting the *groups* around each individual symbol, all exact duplicates are removed to obtain a final list of *groups*. It is clear that the selection of radius r is crucial, as it directly determines the *group* sizes. If chosen too small, no *groups* are formed; on the other hand, a radius too big leads to large and general *groups* and eventually to only one *group* that contains all symbols (and hence emulates the $*$-wildcard). Currently, we rely on manual selection of an appropriate radius. Our initial tests with a k-nearest neighbourhood approach as an alternative to the radius-based approach did not achieve better performance. Further exploration of automatic group selection mechanisms is left for future work.

We extend `SEQL` by first pre-computing *groups* followed by the normal `SEQL` learning procedure. We call this Emb-`SEQL` for **Emb**edding enriched **SEQL**. Once the *groups* are generated, each of them acts as a new base symbol for `SEQL`

and can be part of any k-mer. During the tree search of SEQL, *groups* behave exactly like a normal symbol of the alphabet.

4.2 Semantic Fidelity

Our base assumption is that binary linear classification models (*i.e.,* weighted list of features) are understandable and interpretable as long as their features are. This complies with the *decomposability* propriety of interpretable models proposed by [8]. Nevertheless, determining how interpretable is a set of features is a task often neglected, and still requires manual intervention. To overcome this problem, we propose a *functionally-grounded* protocol [3] based on the *Semantic Fidelity*, a novel metric to measure the interpretability of the features of a linear model for sequence classification without the need for user intervention. Following the rationale that explanations should match user expectations [10], we cast the problem of measuring the interpretability of a set of features to computing distances in the embedding space of an auxiliary background knowledge base. The intuition is that features with positive weights should be highly related to the concept of target class c and in contrast negative features should relate $barc$ the not-target class.

We define the Semantic Fidelity as follows:

$$SF = 1 - \frac{1}{2n} \sum_{\phi_i \in \Phi} h(\phi_i) \tag{1}$$

Φ is the set of features, $\phi_i \in \Phi$ is a feature, n is the number of features, and h is defined as:

$$h(\phi) = |w| \begin{cases} d(\phi, c) & \text{if } w \geq 0 \\ d(\phi, \bar{c}) & \text{otherwise} \end{cases} \tag{2}$$

where c is the positive class of the binary classification task (i.e. the target) and $\bar{c}$ the negative class, w is the weight associated to feature ϕ, and $d(\phi, c)$ is the distance between a k-mer feature ϕ and the concept of the target class c. The distance $d(\phi, c)$ is defined as the average distance between the embeddings $\mathbf{E}_\sigma$ of each individual k-mer symbol $\sigma \in \phi$ and the embedding $\mathbf{e}_c$ of the class:

$$d_c(\phi, c) = \frac{1}{n_\phi} \sum_{\sigma_j \in \phi} \left\| \mathbf{E}_{\sigma_j} - \mathbf{e}_c \right\| \tag{3}$$

where n_ϕ is the number of symbols in ϕ and the embedding $\mathbf{E}_\sigma$ of symbol σ represents a single symbol, or in case of an Emb-SEQL *group* of length n_σ, the average of all symbols τ in the group:

$$\mathbf{E}_\sigma = \begin{cases} \mathbf{e}_\sigma & \text{if } n_\sigma = 1 \\ \dfrac{1}{n_\sigma} \displaystyle\sum_{\tau_k \in \sigma} \mathbf{e}_{\tau_k} & \text{otherwise} \end{cases} \tag{4}$$

We assume the embedding space and weights w to be normalized so that ($w \in [0, 1]$) and the maximum distance $d(\phi, c) = 2$, and consequently $SF \in [0, 1]$, where a higher value means a more interpretable model.

5 Experiments

In this section we assess the interpretability and the predictive power of Emb-SEQL. We experiment in two distinct application scenarios: human activity recognition from wearables, and amino-acid sequences classification. In a second experiment, we show that the background knowledge injection of Emb-SEQL does not affect its predictive power, but improves interpretability.

5.1 Experimental Settings

Datasets. We experiment with a number of symbolic sequence classification datasets and a range of auxiliary background knowledge sources. The symbolic sequence datasets used in the experiments are:

- **OPPORTUNITY (HAR)**: Human activity recognition dataset of wearable sensor data collected from subjects performing actions in a room [17]. It includes inertial measurements from 15 subjects, resulting in 113 sensor recordings provided as multivariate time series. Data points are annotated at different levels of abstraction. For this paper, we aggregate the four low-level labels (*left hand action, left hand object, right hand action, right hand object*) as well as the *locomotion* annotations to form a 5-let. We transform the multidimensional symbolic sequences of OPPORTUNITY by encoding 5-lets into unique symbols (we merge adjacent repeated 5-lets). This procedure results in more than 1,400 unique symbols. We concatenate all records for all subjects, and we window with size 1,000 (roughly 30 seconds) and stride 50. We label a window with its majority class. Our task is predicting the five top-level activities (*Relaxing, Coffee time, Clean up, Sandwich time, Early morning*) from sequences of 5-lets. We use a one-vs-all approach to address the multiclass setting of the dataset. All results are obtained with 10-fold cross validation. Note that for Emb-SEQL we compute the embedding of a *group* by averaging the five embeddings of symbols in a 5-let.
- **Protein**: An excerpt of PhosphoELM[5] used in [23]. It includes sequences of 21 distinct amino acids from the S/T/Y phosphorylation site. Each sequence is labeled with a protein group. We narrow down to two kinase groups (PKA group with 381 sequences and SRC with 157), to compare against the binary classification task results in [23]. We obtained the result by applying 10 fold cross validation as done in [23].

We used the following pre-trained word embeddings:

- **ConceptNet Numberbatch**: ConceptNet Numberbatch 17.06[6] includes word embeddings for more than 1.9M terms from the ConceptNet open data project. It combines data from ConceptNet, word2vec, GloVe, and OpenSubtitles 2016 [20]. Embeddings have dimensionality $k = 300$.
- **GloVe**: these pre-trained embeddings have been created with the GloVe unsupervised model from a large corpus of data crawled from the web [14], and cover 1.9M words. Embeddings have dimensionality $k = 300$.

[5] http://phospho.elm.eu.org/

[6] http://bit.ly/numberbatch

Table 1. The adopted background knowledge graphs

	WordNet	YAGO-41	ChEBI-ChEMBL
Triples	2,429,896	39,198,096	1,947,490
Relations	36	41	46
Entities	1,499,274	8,316,467	938,867

We used the following knowledge graphs (detailed statistics are reported in Table 1):

- **WordNet**: WordNet is a popular lexical database of English terms. Words are grouped into *synsets*, sets of cognitive synonyms that express a distinct concept. Synsets are connected with typed relations that represent conceptual, semantic, and lexical relations. We use the RDF version of WordNet 3.1[7].
- **YAGO-41**: YAGO is a large, broad-scope knowledge graph. We used version 3.1[8]. Due to the large size of YAGO, we only used the following splits: `yagoDBpediaClasses`, `agoDBpediaInstances`, `yagoTaxonomy`, `yagoTypes`, and `yagoFacts`.
- **ChEBI-ChEMBL**: the knowledge graph includes triples from the RDF versions of ChEBI[9] and ChEMBL[10]. ChEBI includes information about small chemical compounds, i.e., molecular entities involved in processes of living organisms. We use the ChEBI-core split (∼1.8M triples). ChEMBL-RDF 24.1 is a manually curated chemical database of bioactive molecules with drug-like properties. We downloaded the splits describing the target triples, and the mappings to ChEBI entities.

For each embedding, we manually select a radius for the *group* generation. The main criteria for the selection are the total number of *groups* as well as their size. The best radii are: GloVe: 0.35, WordNet: 0.185, YAGO-41: 0.23, ConceptNet: 0.23, ChEBI-ChEMBL: 0.65.

Implementation Details. Emb-`SEQL` is implemented in C++. The Semantic Fidelity function is written in Python 3.6. The implementation of the knowledge graph embeddings model uses TensorFlow, on Python 3.6.

Knowledge Graph Embeddings Generation. Besides using pre-trained word embeddings (GloVE and ConceptNet Numberbatch), we also experiment with knowledge graph embeddings. This is done to overcome the single-vector multiple-meaning shortcoming of word embeddings. We learn knowledge graph embeddings for each knowledge graph listed in Table 1. We use ComplEx [21], the neural embedding model that strikes the best trade-off between predictive power and training speed. This is crucial given the size of the knowledge graphs used in the experiments. We rely on typical hyperparameter values known to perform well for splits of WordNet and YAGO: we train the embeddings with dimensionality $k = 150$, AdaGrad optimizer, initial learning rate $\alpha_0 = 0.1$,

[7] `http://wordnet-rdf.princeton.edu/about`
[8] `http://bit.ly/YAGO3`
[9] `https://www.ebi.ac.uk/chebi/`
[10] `http://bit.ly/ChEMBL-RDF`

margin-based pairwise loss function with margin $\gamma = 2$, and negatives per positive ratio $\eta = 2$, $epochs = 100$. Figure 2 shows a PCA-reduced scatterplot of the concept embeddings for the human activity recognition.

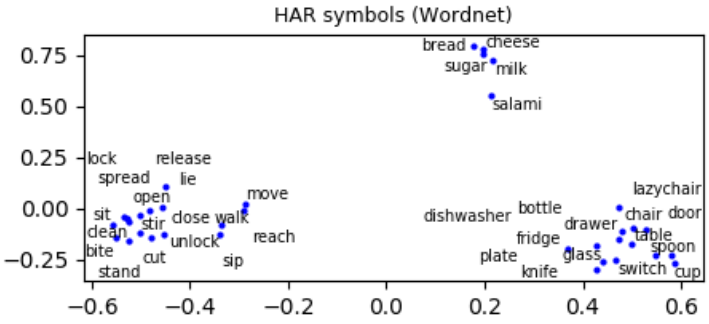

Fig. 2. PCA-reduced plot of the embeddings used in the experiments. Note the clear-cut clusters of concepts (food, kitchenware, verbs).

5.2 Features Interpretability

Human Activity Recognition. We learn features from the OPPORTUNITY dataset with `SEQL` and Emb-`SEQL`, experimenting with word and graph embeddings generated from different auxiliary knowledge bases. We compute the Semantic Fidelity (Equation 1) of the learned features to assess if the features of Emb-`SEQL` obtained with the embedding-driven *groups* are more interpretable than those obtained with plain `SEQL`.

Table 2 reports the semantic fidelity SF obtained by five binary classifiers defined for each of the five top-level activities to predict. Results are stable across the five target classes. Emb-`SEQL` outperforms `SEQL` with most of the embeddings: WordNet brings 4.6% increase in Semantic Fidelity, while GloVe obtains a 2.3% increase and ConceptNet a 0.4% increase. YAGO-41, on the other hand does not bring any advantage over plain `SEQL`. This is probably due to sparse relations and lack of redundancy in the YAGO splits we used to build YAGO-41. Future experiments will use a complete version of YAGO. Figure 3 shows an example of the embedded features of Emb-`SEQL` and `SEQL` in a PCA-reduced representation.

Amino Acids. We also experiment with the Protein dataset. As for the HAR scenario, we learn features with both `SEQL` and Emb-`SEQL`. For Emb-`SEQL` we use the ChEBI-ChEMBL auxiliary knowledge base, which we inject in the model as knowledge graph embeddings. Table 2 reports the Semantic Fidelity SF over a single class, as this is a binary classification task: Emb-`SEQL` reaches a Semantic Fidelity 1.5% higher than its counterpart, thus making its features more interpretable.

Table 2. Semantic Fidelity for the human activity recognition (HAR) and Protein sequence classification tasks. Higher scores indicates more interpretable models. For the HAR experiment we report the mean semantic fidelity $\overline{SF}$ obtained by five binary classifiers defined for each of the five top-level activities to predict. We also report the results for each individual class.

HAR:								
Embeddings	Model	$\overline{SF}$	std	Class 1	Class 2	Class 3	Class 4	Class 5
GloVe	SEQL	0.902	0.028	0.930	0.871	0.925	0.865	0.921
	Emb-SEQL	**0.923**	0.025	0.958	0.888	0.931	0.901	0.938
ConceptNet	SEQL	0.871	0.033	0.908	0.828	0.895	0.833	0.889
	Emb-SEQL	**0.875**	0.025	0.903	0.853	0.887	0.836	0.894
YAGO-41	SEQL	**0.867**	0.029	0.899	0.847	0.893	0.823	0.872
	Emb-SEQL	0.835	0.043	0.897	0.824	0.861	0.767	0.827
WordNet	SEQL	0.894	0.025	0.921	0.879	0.918	0.857	0.895
	Emb-SEQL	**0.936**	0.010	0.937	0.945	0.943	0.917	0.939
Protein:								
Embeddings	Model	$\overline{SF}$						
ChEBI-ChEMBL	SEQL	0.708						
	Emb-SEQL	**0.719**						

5.3 Classification Quality

Human Activity Recognition. Besides interpretability, we are also interested in assessing whether Emb-`SEQL` achieves a predictive power comparable to `SEQL`. Therefore, we compare the performance of Emb-`SEQL` to `SEQL` without any knowledge injection, as well as to a `SVM` baseline, and a LSTM-based neural network. We use a `SVM` with RBF kernel implemented with libsvm [2]. We extracted all k-mers up to $k = 6$ and explicitly generated the feature vector representation for each example as input for the SVM. The LSTM has 64 hidden units followed by a single 16-unit hidden layer classifier which maps to the number of output classes. The model is trained with Adam on weighted cross entropy loss, to mitigate class imbalance. It is trained for 100 epochs with early stopping.

Table 3 shows the results of the experiment on the OPPORTUNITY dataset with the above mentioned embeddings. We show the weighed F1 score, as well as the accuracy, excluding the null class (as done in prior work). Results show that all `SEQL` models, regardless of the injection of auxiliary knowledge, outperform the `SVM` in both metrics, as well as the LSTM. The performance of Emb-`SEQL` is comparable to `SEQL`. We conclude that the injection of knowledge into Emb-`SEQL` did not hurt the performance of the model with regard to Accuracy, but lead to better model interpretability (according to Semantic Fidelity).

Amino Acids. A similar conclusion can be drawn for the evaluation on the Protein dataset. Table 3 shows the F1 score and accuracy of `SEQL` and Emb-`SEQL` with ChEBI-ChEMBL embeddings as well as for the LSTM-based architecture described for HAR and SCIS_MA (sequence classification based on association rules) and its HMM baseline [23]. It is clearly visible that the Accuracy of `SEQL`

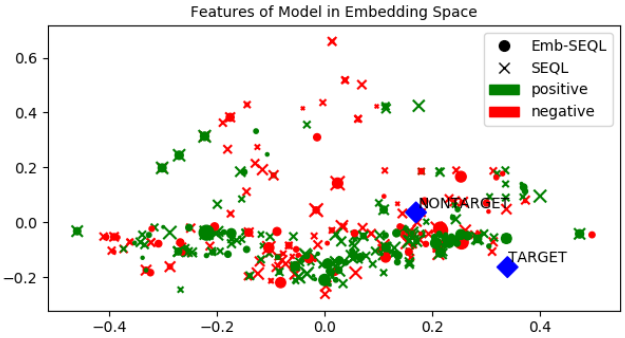

Fig. 3. PCA-reduced plot of the Emb-`SEQL` and `SEQL` models for class *coffee time* in the WordNet embedding space. Note positive (negative) weighted features of Emb-`SEQL` are closer to the TARGET (NONTARGET) class concepts.

Table 3. Results on the HAR dataset (OPPORTUNITY) and Protein dataset. Best results in bold.

Dataset	Model	Embeddings	F1	Accuracy
HAR	SVM		0.502	0.564
	LSTM		0.767	0.810
	SEQL		**0.973**	**0.961**
	Emb-SEQL	ConceptNet	0.965	0.951
		GloVe	0.961	0.945
		WordNet	0.968	0.955
		YAGO-41	0.957	0.941
Protein	SCIS_MA		-	**0.948**
	HMM		-	0.918
	LSTM		0.797	0.796
	SEQL		**0.902**	0.903
	Emb-SEQL	ChEBI-ChEMBL	0.898	0.901

and Emb-`SEQL` lacks somewhat behind SCIS_MA and HMM, but the injection of knowledge doesn't significantly hurt the performance of Emb-`SEQL`.

6 Conclusion

We show that semantic embeddings help generate more interpretable features for sequence classification with linear models. Besides, we also show that distances in embedding spaces can be used to quantify how interpretable such features are. Future work will include exploration of different clustering techniques to form groups in Emb-`SEQL`. An important axis of work will be validating the Semantic Fidelity against human-grounded and application-grounded evaluation protocols. Furthermore, we will investigate the application of the Semantic Fidelity to other feature-based models.

References

1. Alipanahi, B., Delong, A., Weirauch, M.T., Frey, B.J.: Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nature Biotechnology **33**(8), 831–838 (2015)
2. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011)
3. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
4. Freitas, A.A.: Comprehensible classification models: a position paper. ACM SIGKDD explorations **15**(1), 1–10 (2014)
5. Gsponer, S., Smyth, B., Ifrim, G.: Efficient sequence regression by learning linear models in all-subsequence space. In: Procs of ECML-KDD. pp. 37–52 (2017)
6. Ifrim, G., Wiuf, C.: Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. Procs of SIGKDD (2011)
7. Leslie, C., Eskin, E., Noble, W.S.: The spectrum kernel: a string kernel for svm protein classification. Procs of the Pacific Symp on Biocomputing **7**, 564–575 (2002)
8. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3) (Jun 2018)
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
10. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence (2018)
11. Minervini, P., Demeester, T., Rocktäschel, T., Riedel, S.: Adversarial sets for regularising neural link predictors. In: Procs of UAI (2017)
12. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Procs of the IEEE **104**(1), 11–33 (2016)
13. Nutini, J., Schmidt, M., Laradji, I.H., Friedlander, M., Koepke, H.: Coordinate descent converges faster with the gauss-southwell rule than random selection. In: Procs of ICML. vol. 37, pp. 1632–1641 (2015)
14. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Procs of EMNLP. pp. 1532–1543 (2014)
15. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. Procs of the IEEE **77**(2), 257286 (1989)
16. Rocktäschel, T., Singh, S., Riedel, S.: Injecting logical background knowledge into embeddings for relation extraction. In: HLT-NAACL. pp. 1119–1129 (2015)
17. Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Frster, K., Trster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Sagha, H., Bayati, H., Creatura, M., d. R. Milln, J.: Collecting complex activity datasets in highly rich networked sensor environments. In: INSS. pp. 233–240 (June 2010)
18. Sonnenburg, S., Rätsch, G., Schäfer, C.: Learning interpretable SVMs for biological sequence classification. Research in Computational Molecular Biology (2005)
19. Sonnenburg, S., Rätsch, G., Rieck, K.: Large scale learning with string kernels. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines, pp. 73–103. MIT Press, Cambridge, MA. (2007)
20. Speer, R., Chin, J., Havasi, C.: ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In: Procs of AAAI. pp. 4444–4451 (2017)
21. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: procs of ICML. pp. 2071–2080 (2016)
22. Xing, Z., Pei, J., Keogh, E.J.: A brief survey on sequence classification. SIGKDD Explorations **12**(1), 40–48 (2010)
23. Zhou, C., Cule, B., Goethals, B.: Pattern based sequence classification. IEEE Transactions on Knowledge and Data Engineering **28**(5), 1285–1298 (2016)
24. Zhou, J., Troyanskaya, O.G.: Predicting effects of noncoding variants with deep learning based sequence model. Nature Methods **12**(10), 931934 (Aug 2015)