Supplemental Materials

Term frequency–inverse document frequency

Term frequency-inverse document frequency (TFIDF)[1] measures how important a term is to a document in a corpus. It is often used to provide weighted representations of documents for information retrieval. Such metric can be factored into term-frequency (TF) and inverse document frequency (IDF). Binary TF measures the existence of a term in a document, and is given in the following equation:

$$tf(t,d) = \begin{cases} 1, & \text{if } t \text{ occurs in a document} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where t is the term and d is the document, here we sum the value over the corpus. IDF is defined as the logarithm of the inverse fraction of the number of documents that contain the term t, as expressed in the following equation:

$$idf(t,D) = \log(\frac{|D|}{|\{d|t \in d, d \in D\}|}), \tag{2}$$

where D represents the corpus, i.e. a set of documents d; the denominator is the number of documents in which term t appears; and the $|\cdot|$ denotes set size. IDF assigns larger values to terms that occur less frequently across documents in the corpus. It is usually employed to find terms that do not occur frequently but are considered meaningful in documents.

Finally, combining TF and IDF scores, the tf_idf value is calculated in Equation 3 as:

$$tf\_idf(t,d,D) = tf(t,d) \times idf(t,D). \tag{3}$$

Considering base pairs as terms, we will use this method to score each base pair in a region sets and select top-scoring regions for further process.

Word Embedding

Traditional approaches in NLP represented text as a sparse vector with the length corresponding to the size of the vocabulary. Named one-hot representation, this method creates a vector of zeros and ones representing word occurrence in the vocabulary. Alternatively, the values of the vector can be the number of word occurrences in the document. Distributed word representation, generally known as word embedding, is used to solve the problems of high dimensionality and sparsity in the representation. In this representation, each word is described by the surrounding context[2] which contains semantic and syntactic information about the word. A language modeling task is employed to construct such representations. Distributed representation learning is first introduced by Hinton[3] and is developed as a language modeling concept by Bengio[4].

Word2vec[5] is a series of methods to represent a word using a numerical vector. The main idea of building this representation is to express a word by its context by training a shallow neural network. Each row of the weight matrix of the neural network represents a word vector. The neural network takes in a one-hot encoded word as its input, then passes it through hidden layers to get an output that predicts the word, given its context. The weights of the hidden layer are updated if the words share the same context. After the training, the hidden layer weights represent a vector for each word. In our approach, we adapt word2vec to genomic region sets and use the trained model to extract region embedding vectors, which we then combine to build region set embeddings.

Run-time analysis

There are two steps in learning representations for a dataset. 1) building the representation models on the training data and 2) transforming the new test set to the target representations. Regarding run-time analysis, in the model building phase on the training data, word2vec does require more time to train the neural network, but after the training, we can simply use and update the model. The training time depends on hyperparameters such as dimension size, number of documents, and vocabulary size. For example, on average it takes 45 minutes for 50-dimension vectors and 140 minutes for 100-dimension vectors on our data. The union and tf_idf representation use 100 files to build the feature set and in fact there is no training. Therefore, the run-time to build these models are faster than training the word2vec model (Table S4).

However, in the second step for transforming new test dataset, for the union and tf_idf representations, a region-set needs to be mapped to the high dimensional space and then use the saved PCA model to reduce the dimension to 100 while region-set embedding is calculated in low dimension. Transforming a dataset of 695 files to the representation vectors takes around 1 minute for the region-set2vec model and around 2.5 minutes to transform and reduce the dimension for union representation (Table S5).

SVM and PCA kernel analysis

We selected the kernel for the SVM classifier based on the cross-validation results on the training dataset. We split the training data into 10 folds to choose the best hyperparameters using the cross validation score. *Linear*, *rbf*, and *polynomial* kernel were employed and linear kernel was chosen due to the higher average of the scores across all folds. The same kernels were used for PCA dimension reduction and *Linear* kernel achieved the best results (Table S6, Table S7, and Table S8).

UMAP parameter analysis

To study the effect of UMAP dimension reduction parameters, we used different values to generate various two-dimension plots (Table S9). We found that using 100 (50, 100, 150, and 200 were tested) neighbors is a good choice to produce sensible and robust visualizations when changing the other hyperparameter. The distinction between the final plots for different values of minimum distance was negligible. With the 100 neighbors in the UMAP configuration, we chose Euclidean, Cosine, Jaccard, and Dice as the similarity metrics with Jaccard and Dice used specifically for the binary representations. We observed that 2-cluster structure is evident in the Euclidean distance plots of both the union and tf_idf representations, although in the Jaccard and Dice plots, this structure still exists but it is less obvious. We conjectured that the 2-cluster structure shown in the Euclidean distance plots is due to the following two reasons. Since a binary representation is robust to small perturbations, such as a small percentage of dropping or shifting, after the perturbation rate exceeds a certain level, the Euclidean distance will change significantly, and the UMAP captures the small change and big change as two distinct clusters in the reduced dimension space. Moreover, since the other distance metrics are normalized versions of the raw change with a normalization constant not reflecting the perturbation rates, their plots cannot produce distinctive clusters. Compared to the binary representations, the region-set2vec representation is able to reflect gradual changes under both the Euclidean and Cosine distance metrics (Fig. S1a-d).

To investigate the effect of each aspect of perturbation on the final representation, we generated three different datasets by varying one aspect of perturbation at a time, add, drop, and shift. Then we plot the new datasets for exploration. First, we randomly add regions to the sample file to create perturbation (Fig. S2a). Dropping regions from the file had the same effect on the final representations (Fig. S2b). By shifting the regions some of the regions drop out of the regions in the universe and some overlap the regions in the universe of possible regions (Fig. S2c). Increasing the rate of perturbation on this aspect gradually change the region-set embeddings while there are no obvious pattern in other representations. This indicates that the similarity between the original file and the perturbed file is preserved in the region-set2vec representation rather than the other methods on every aspect of perturbation.

Supplementary figures

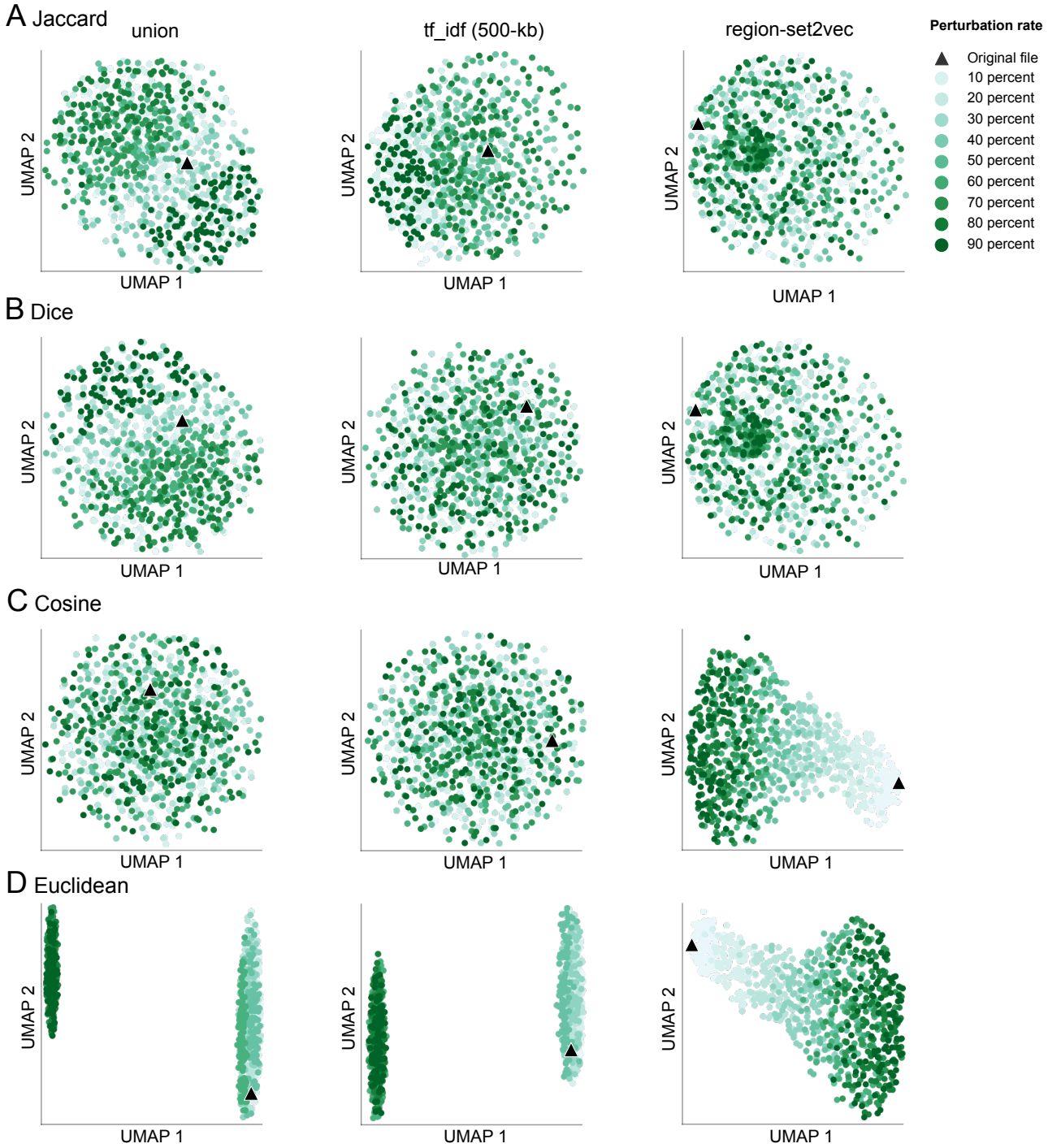

***Figure S1*: UMAP visualization with different similarity metrics for simulated dataset using three representation methods.** *(**a**). Jaccard (**b**). Dice (**c**). Cosine (reproduced from Figure 4) (**d**). Euclidean. Jaccard and Dice distance were poor metrics for all three representations, while cosine and Euclidean distance show that region-set2vec has a better ability to capture similarity.*

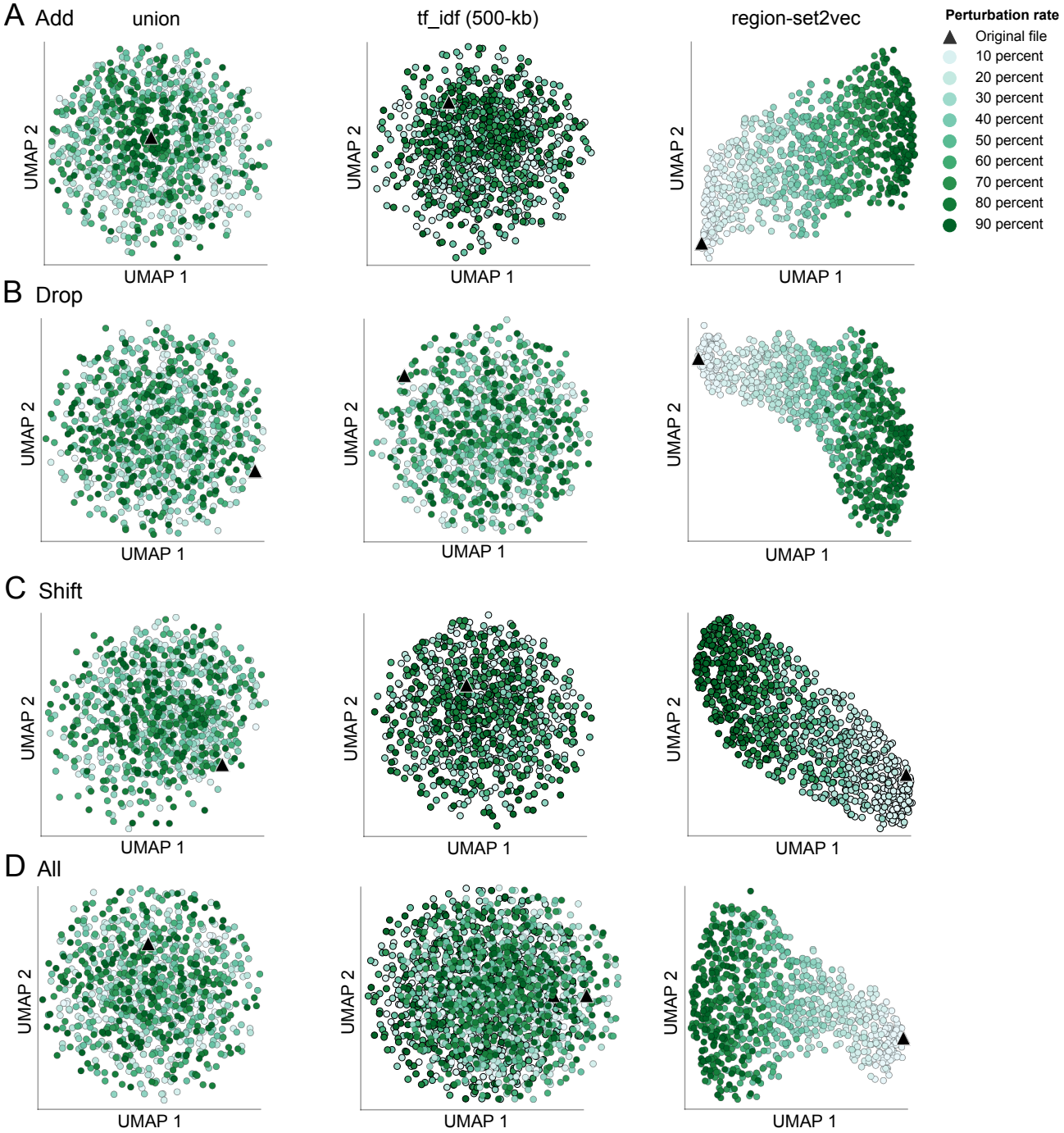

Figure S2*: *UMAP visualization of the simulated dataset with perturbations made individually on each of the three representation methods. Cosine is used as the distance metric in UMAP method. *(**a**). Adding regions to each file. (**b**). Dropping regions from each file. (**c**). Shifting regions in each file. (**d**). Combination of all three types of perturbation (reproduced from Figure 4). When adding and shifting regions, it appears that union and tf_idf representations do not capture similarity, as all of the points spread out without any pattern. However, the region-set2vec representation indicates more perturbed files are more distant than less perturbed files. When only dropping regions, the UMAP plots look similar to when all perturbations are present, because dropping regions is the only perturbation guaranteed to make changes to the vector representations.*

Supplementary tables

	Antibody	Cell line	Tissue
# regions	136,284	180,425	150,681
median region length	393	462	428
mean region length	761	1,057	801

Table S1: Union representation properties

	Antibody	Cell line	Tissue
1	0.9209	0.9547	0.8932
2	0.9180	0.9518	0.9045
3	0.9194	0.9598	0.9023
4	0.9122	0.9591	0.9000
5	0.9165	0.9591	0.9205
6	0.9122	0.9572	0.9023
7	0.9180	0.9561	0.8932
8	0.9122	0.9581	0.9000
9	0.9165	0.9494	0.9114
10	0.9180	0.9568	0.8932
11	0.9151	0.9477	0.9114
12	0.9137	0.9485	0.9068
13	0.9094	0.9522	0.8977
14	0.9209	0.9583	0.9136
15	0.9137	0.9575	0.9091
16	0.9180	0.9551	0.9136
17	0.9122	0.9498	0.9045
18	0.9089	0.9527	0.9091
19	0.9201	0.9469	0.9000
20	0.9187	0.9569	0.9023

Table S2: SVM-PCA classifier performance robustness test. We tested 20 different universes created from different sets of 100 random BED files to confirm that the random selection did not affect the classifier performance.

Representation method	Antibody
union representation	0.9317
tf_idf - 1000-kb	0.9101
tf_idf - 500-kb	0.9317
tf_idf - 100-kb	0.8885
region-set2vec embedding-averaging	**0.9568**
doc2vec	0.5606

Table S3: Comparison on SVM classifier performance on averaging and doc2vec combination function

Representation method	Building models
union representation	00:03:00
tf_idf	00:09:00
region-set2vec embedding	02:20:00

Table S4: Run-time to build the representation models

Representation method	Vectorization	PCA	Total
union representation	0:01:48	0:00:27	0:02:15
tf_idf_1000-kb	0:00:58	0:00:12	0:01:10
tf_idf_500-kb	0:00:50	0:00:08	0:00:58
tf_idf_100-kb	0:00:15	0:00:03	0:00:18
region-set2vec embedding	0:01:04		0:01:04

Table S5: Run-time to transform test dataset to representations for downstream tasks

Representation method	Antibody			Cell line			Tissue		
	linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
union representation	0.9255	0.7681	0.7523	0.9470	0.5318	0.6984	0.8462	0.5710	0.5274
tf_idf_1000-kb	0.9154	0.7865	0.7807	0.9305	0.5475	0.5214	0.8413	0.6248	0.5732
tf_idf_500-kb	0.9129	0.7897	0.7857	0.9050	0.5614	0.5300	0.8402	0.6803	0.6063
tf_idf_100-kb	0.8923	0.7490	0.7476	0.8690	0.5716	0.5546	0.8435	0.6919	0.6622
region-set2vec embedding	**0.9445**			**0.9622**			**0.8747**		

Table S6: SVM classifier performance with linear kernel

Representation method	Antibody			Cell line			Tissue		
	linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
union representation	0.8592	0.8451	0.8606	0.5111	0.4869	0.6850	0.7729	0.7696	0.7746
tf_idf_1000-kb	0.8174	0.8088	0.8196	0.5370	0.4674	0.5527	0.7740	0.7347	0.7757
tf_idf_500-kb	0.8088	0.8027	0.8109	0.5171	0.4194	0.5381	0.7751	0.7185	0.7768
tf_idf_100-kb	0.7976	0.7612	0.8016	0.4892	0.3964	0.5567	0.7817	0.7615	0.7857
region-set2vec embedding	**0.9237**			**0.9236**			**0.8328**		

Table S7: SVM classifier performance with RBF kernel

Representation method	Antibody			Cell line			Tissue		
	linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
union representation	0.8167	0.8027	0.8239	0.4829	0.5495	0.7049	0.6253	0.6174	0.6275
tf_idf_1000-kb	0.8023	0.7911	0.8077	0.7362	0.6946	0.7425	0.5973	0.5911	0.6484
tf_idf_500-kb	0.8023	0.7875	0.8073	0.7232	0.6798	0.7342	0.6007	0.5827	0.6669
tf_idf_100-kb	0.8063	0.7623	0.8142	0.7176	0.6593	0.7246	0.6633	0.5673	0.7071
region-set2vec embedding	**0.8790**			**0.9025**			**0.7494**		

Table S8: SVM classifier performance with polynomial kernel

Hyperparameter	Values
number of neighbors	50, 100, 150, 200
minimum distance	0.1, 0.01, 0.05
distance metric	Euclidean, Cosine, Jaccard, Dice

Table S9: Values of the hyperparameters

References

1. Salton, G. & Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* **24**, 513–523 (1988).

2. Firth, J. R. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis* (1957).

3. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323**, 533–536 (1986).

4. Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C. A neural probabilistic language model. *Journal of Machine Learning Research* **3**, 1137–1155 (2003).

5. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. in *Advances in neural information processing systems* 3111–3119 (2013).