

Chapter 2

Methods

We previously introduced information-theoretic metrics for evaluating classification performance in protein function prediction which we describe here [2]. In this learning scenario, the input space \mathcal{X} represents proteins, whereas the output space \mathcal{Y} contains directed acyclic graphs describing protein function according to the Gene Ontology (GO).

Because of the hierarchical nature of GO, both experimental and computational annotations need to satisfy the *consistency requirements*:

- i. If vertex (term) v from the ontology is true, then all of its ancestors must also be true.
- ii. If vertex (term) v from the ontology is false, then all of its descendants must also be false.

By enforcing these requirements, we frame the task of a classifier as assigning the best consistent subgraph of the ontology to each new protein and output a prediction score for this subgraph and/or each predicted term.

We simplify the exposition by referring to such graphs as prediction or annotation graphs. In addition, we frequently treat consistent graphs as sets of nodes or functional terms and use set operations to manipulate them.

We now proceed to provide a definition for the information content of a (consistent) subgraph in the ontology. Then, using this definition, we derive information-theoretic performance evaluation metrics for comparing pairs of graphs.

2.1 Calculating the Joint Probability of a Graph

Let each term in the ontology be a binary random variable and consider a fixed but unknown probability distribution over \mathcal{X} and \mathcal{Y} according to which the quality of a prediction process will be evaluated. We shall assume that the prior distribution of a target can be factorized according to the structure of the ontology, i.e., we assume a Bayesian network as the underlying data generating process for the target

variable. According to this assumption, each term is independent of its ancestors given its parents and, thus, the full Joint probability can be factorized as a product of individual terms obtained from the set of Conditional probability tables associated with each term [7],

$$\Pr(\mathbf{V}) = \prod_{v \in \mathbf{V}} \Pr(v|\mathcal{P}(v)), \quad (2.1)$$

where \mathbf{V} denotes all vertices the ontology, v denotes a vertex in \mathbf{V} and $\mathcal{P}(v)$ is the set of parent nodes of v . Here we use \mathbf{V} (all terms) instead of T (only true terms) to illustrate the fact that we are calculating the probability of a given configuration of the full ontology with the inclusion of all true and false terms. In practice the true terms, or the subgraph T , are generally all that is considered both in the context of this chapter and in other papers addressing similar topics.

The scope of terms considered can be reduced when calculating the joint probability of a configuration of \mathbf{V} without affecting the final probability value. Because of the enforced consistency requirement (i.e., all ancestors of true terms are true; all descendants of false terms are false) the full joint probability of a configuration of the ontology, \mathbf{V} , can be calculated by considering only terms whose parents are all true. Equation 2.1 can be rewritten as

$$\Pr(\mathbf{V}) = \prod_{v \in T \cup \mathcal{C}(T)} \Pr(v|\mathcal{P}(v)), \quad (2.2)$$

where T denotes all terms in \mathbf{V} that are true, $\mathcal{C}(T)$ defines false terms all of whose parents are true (children of leaf terms in T), and $T \cup \mathcal{C}(T)$ defines the union of the two sets.

The derivation of Eq. 2.2 can be obtained by considering Fig. 2.1 which illustrates a sample ontology with 9 terms and Table 2.1 which represents the conditional probability distribution, or table, for vertex g . If we write the joint probability of \mathbf{V} using Eq. 2.1 we obtain

$$\begin{aligned} \Pr(\mathbf{V}) = & \Pr(a = 1) \times \Pr(b = 0|a = 1) \times \Pr(c = 1|a = 1) \times \Pr(d = 0|b = 0) \\ & \times \Pr(e = 1|c = 1) \times \Pr(f = 0|c = 1) \times \Pr(g = 0|d = 0, e = 1) \\ & \times \Pr(h = 0|f = 0) \times \Pr(i = 0|f = 0). \end{aligned}$$

First, we see that terms h and i contribute nothing to the joint probability because the probability a term is false given its parents are all false is always equal to 1. Although b and f are false, because all of their parents are true the resulting probability is not 1. Term g can also be ignored because one of its parents is false as illustrated by considering Table 2.1. The resulting joint probability can subsequently be rewritten as

$$\begin{aligned} \Pr(\mathbf{V}) = & \Pr(a = 1) \times \Pr(b = 0|a = 1) \times \Pr(c = 1|a = 1) \\ & \times \Pr(e = 1|c = 1) \times \Pr(f = 0|c = 1). \end{aligned}$$

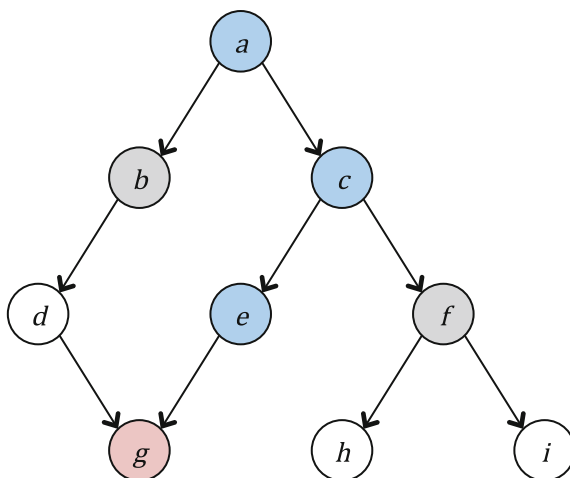


Fig. 2.1 A directed acyclic graph representing relationships between terms in a potential ontology. True terms, or T , are represented by *blue* colored vertices: $\{a, c, e\}$. The children of true terms, $\mathcal{C}(T)$, are denoted by *grey* and *light-red* vertices: $\{b, f, g\}$. Terms which do not contribute anything to the full joint probability are shown in *white* and *light-red*: $\{d, g, h, i\}$. Term g is colored *light red* to point out a special case when calculating the full joint probability of a graph. Although one of term g 's parents are true, because it has one false parent it will not contribute to the calculation of the full joint probability because the probability it is false is always equal to one (Table 2.1). All *blue* and *grey* terms taken together comprise the *markov blanket* of T , and are necessary when calculating the full joint probability of a configuration, or a combination of true and false values for all terms, in \mathbf{V}

Table 2.1 A table showing a conditional probability distribution for term g from Fig. 2.1

d	e	$\Pr(g)$	
		True	False
True	True	.7	.3
True	False	0	1
False	True	0	1
False	False	0	1

The purpose of this table is to illustrate how the consistency requirement affects joint probability tables in the Bayesian network. If at least one of a term's parents is false it is guaranteed that that term is also false and contributes nothing. This is shown by considering the last three rows in the table

Because in practice most annotations only draw upon a small fraction of terms, the vast majority of terms will be negative and will not contribute to the calculation of a given joint probability. For this reason, although calculating the full joint probability has an asymptotic upper bound defined by the number of terms in the ontology, in practice it should have much lower complexity.

In this context, we are only interested in marginal probabilities that a protein is experimentally associated with a consistent subgraph T in the ontology. This probability can be expressed as

$$\Pr(T) = \prod_{v \in T} \Pr(v|\mathcal{P}(v)). \quad (2.3)$$

Equation 2.3 can be derived from the full joint factorization by first marginalizing over the leaves of the ontology and then moving toward the root(s) for all nodes not in T .

Although marginalizing over all negative terms gives a formulation that deviates from calculating the full joint probability it both simplifies calculating the information content of a subgraph and can be philosophically justified. Unobserved annotations are only putative negative observations because explicit negative annotations are rarely deposited in biological databases and in ignoring them we are implicitly recognizing them as such.

2.1.1 Calculating the Information Content of a Graph

Now that we have properly defined the joint probability of a set of terms, or subgraph of the ontology, calculating the information content of those terms is relatively straightforward. The information content of a subgraph can be thought of as the number of bits of information one would receive about a protein if it were annotated with that particular subgraph. We calculate the information content of a subgraph T in a straightforward manner as

$$i(T) = \log \frac{1}{\Pr(T)}$$

and use a base 2 logarithm as a matter of convention. The information content of a subgraph T can now be expressed by combining the previous two equations as

$$i(T) = \sum_{v \in T} \log \frac{1}{\Pr(v|\mathcal{P}(v))} \quad (2.4a)$$

$$= \sum_{v \in T} ia(v), \quad (2.4b)$$

where, to simplify the notation, we use $ia(v)$ to represent the negative logarithm of $\Pr(v|\mathcal{P}(v))$. Term $ia(v)$ can be thought of as the increase, or accretion, of information obtained by adding a child term to a parent term, or set of parent terms, in an annotation. We will refer to $ia(v)$ as *information accretion* (perhaps information gain would be a better term, but because it is frequently used in other applications to describe an expected reduction in entropy, we avoid it in this situation).

A simple ontology containing five terms together with a conditional probability table associated with each node is shown in Fig. 2.2a. Note that because of the graph consistency requirement, each conditional probability table is limited to a single

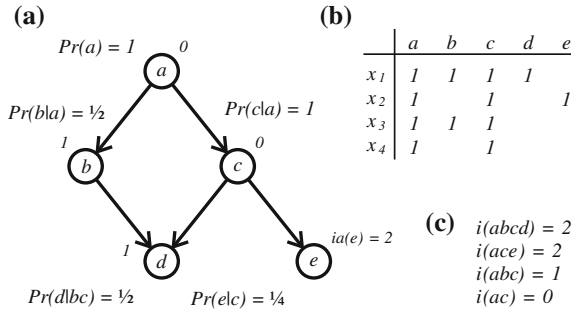


Fig. 2.2 An example of an ontology, data set, and calculation of information content. **a** An ontology viewed as a Bayesian network together with a conditional probability table assigned to each node. Each conditional probability table is limited to a single number due to the consistency requirement in assignments of protein function. Information accretion calculated for each node, e.g., $ia(e) = -\log \Pr(e|c) = 2$, are shown in grey next to each node. **b** A data set containing four proteins whose functional annotations are generated according to the probability distribution from the Bayesian network. **c** The total information content associated with each protein found in panel **b**; e.g., $i(ace) = ia(a) + ia(c) + ia(e) = 2$. Note that $i(ab) = 1$ and $i(abcde) = 4$ although proteins with such annotation have not been observed in panel **b**

number. For example, at node b in the graph, the probability $\Pr(b = 1|a = 1)$ is the only one necessary because $\Pr(b = 0|a = 1) = 1 - \Pr(b = 1|a = 1)$ and because $\Pr(b = 1|a = 0)$ is guaranteed to be 0. In Fig. 2.2b we show a sample data set of four proteins functionally annotated according to the distribution defined by the Bayesian network. In Fig. 2.2c, we show the total information content for each of the four annotation graphs.

2.1.2 Comparing Two Annotation Graphs

We now consider a situation in which a protein's true and predicted function are represented by graphs T and P , respectively. We define two metrics that can be thought of as the information-theoretic analogs of recall and precision, and refer to them as remaining uncertainty and misinformation, respectively.

Definition 1. The *remaining uncertainty* about the protein's true annotation corresponds to the information about the protein that is not yet provided by the graph P . More formally, we express the remaining uncertainty (ru) as

$$ru(T, P) = \sum_{v \in T \setminus P} ia(v), \quad (2.5)$$

which is simply the total information content of the nodes in the ontology that are contained in true annotation T but not in the predicted annotation P . Note that, in

a slight abuse of notation, we apply set operations to graphs to manipulate only the vertices of these graphs.

Definition 2. The *misinformation* introduced by the classifier corresponds to the total information content of the nodes along incorrect paths in the prediction graph P . More formally, the misinformation is expressed as

$$mi(T, P) = \sum_{v \in P \setminus T} ia(v), \quad (2.6)$$

which quantifies how misleading a predicted annotation is.

Here, a perfect prediction (one that achieves $P = T$) leads to $ru(T, P) = mi(T, P) = 0$. However, both $ru(T, P)$ and $mi(T, P)$ can be infinite in the limit. In practice, though, $ru(T, P)$ is bounded by the information content of the particular annotation, while $mi(T, P)$ is only limited by the number of annotations a predictor chooses to return.

To illustrate calculation of remaining uncertainty and misinformation, in Fig. 2.2 we show a sample ontology where the true annotation of a protein T is determined by the two leaf terms t_1 and t_2 , whereas the predicted subgraph P is determined by the leaf terms p_1 and p_2 . The remaining uncertainty $ru(T, P)$ and misinformation $mi(T, P)$ can now be calculated by adding the information accretion corresponding to the nodes circled in grey.

Finally, this framework can be used to define the semantic similarity between the protein's true annotation and the predicted annotation without relying on identifying an individual common ancestor between pairs of leaves (this node is usually referred to as the most informative common ancestor; [4]) (Fig. 2.3). The information content of the subgraph shared by T and P is one such possibility; i.e.,

$$s(T, P) = \sum_{v \in T \cap P} ia(v).$$

2.1.3 Measuring the Quality of Function Prediction

A typical predictor of protein function usually outputs scores that indicate the strength (e.g., posterior probabilities) of predictions for each term in the ontology. To address this situation, the concepts of remaining uncertainty and misinformation need to be considered as a decision threshold function of a τ . In such a scenario, predictions with scores greater than or equal to τ are considered positive predictions, while the remaining associations are considered negative (if the strength of a prediction is expressed via P-values or e-values, values lower than the threshold would indicate positive predictions). Regardless of the situation, every decision threshold results in a separate pair of values corresponding to the remaining uncertainty $ru(T, P(\tau))$ and misinformation $mi(T, P(\tau))$.

$$s(T, P) = \sum_{v \in T \cap P} ia(v).$$

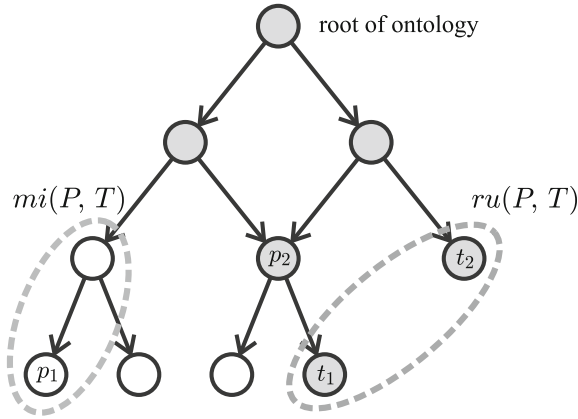


Fig. 2.3 Illustration of calculating remaining uncertainty and misinformation given a predicted annotation graph P and a graph of true annotations T . Graphs P and T are uniquely determined by the leaf nodes p_1 , p_2 , t_1 , and t_2 , respectively. Nodes colored in grey represent graph T . Nodes circled in grey are used to determine remaining uncertainty (ru ; right side) and misinformation (mi ; left side) between T and P

The remaining uncertainty and misinformation for a previously unseen protein can be calculated as expectations over the data generating probability distribution. Practically, this can be performed by averaging over the entire set of proteins used in evaluation, i.e.,

$$ru(\tau) = \frac{1}{n} \sum_{i=1}^n ru(T_i, P_i(\tau)) \quad (2.7)$$

and

$$mi(\tau) = \frac{1}{n} \sum_{i=1}^n mi(T_i, P_i(\tau)) \quad (2.8)$$

where n is the number of proteins in the data set, T_i is the true set of terms for protein x_i , and $P_i(\tau)$ is the set of predicted terms for protein x_i given decision threshold τ . Note that once the set of terms with scores greater than or equal to τ is determined, the set $P_i(\tau)$ is composed of the unique union of the ancestors of all predicted terms. As the decision threshold is moved from its minimum to its maximum value, the pairs of $(ru(\tau), mi(\tau))$ will result in a curve in 2D space. We refer to such a curve using $(ru(\tau), mi(\tau))_\tau$. Removing the normalizing constant ($\frac{1}{n}$) from the above equations

would result in the total remaining uncertainty and misinformation associated with a database of proteins and a set of predictions.

2.1.4 Weighted Metrics

One disadvantage of definitions in Eqs. 2.7 and 2.8 is that an equal weight is given to proteins with low and high information content annotations when averaging. To address this, we assign a weight to each protein according to the information content of its experimental annotation. This formulation naturally downweights proteins with less informative annotations compared to proteins with rare, and therefore more informative (surprising), annotations. In biological data sets, frequently seen annotations have a tendency to be incomplete or shallow annotation graphs and arise due to the limitations or high-throughput nature of some experimental protocols. We define *weighted remaining uncertainty* as

$$wru(\tau) = \frac{\sum_{i=1}^n i(T_i) \cdot ru(T_i, P_i(\tau))}{\sum_{i=1}^n i(T_i)} \quad (2.9)$$

and *weighted misinformation* as

$$wmi(\tau) = \frac{\sum_{i=1}^n i(T_i) \cdot mi(T_i, P_i(\tau))}{\sum_{i=1}^n i(T_i)} \quad (2.10)$$

2.1.5 Semantic Distance

Finally, to provide a single performance measure which can be used to rank and evaluate protein function prediction algorithms, we introduce *semantic distance* as the minimum distance from the origin to the curve $(ru(\tau), mi(\tau))_\tau$. More formally, the semantic distance S_k is defined as

$$S_k = \min_{\tau} (ru^k(\tau) + mi^k(\tau))^{\frac{1}{k}}, \quad (2.11)$$

where k is a real number ≥ 1 . Setting $k = 2$ results in the minimum Euclidean distance from the origin. The preference for Euclidean distance ($k = 2$) over say Manhattan distance ($k = 1$) is to penalize unbalanced predictions with respect to the depth of predicted and experimental annotations.

2.1.6 Precision and Recall

In order to contrast the semantic distance-based evaluation with more conventional performance measures, in this section we briefly introduce precision and recall for measuring functional similarity. As before, we consider a set of propagated experimental terms T and predicted terms $P(\tau)$, and define precision as the fraction of terms predicted correctly. More specifically,

$$pr(T, P(\tau)) = \frac{|T \cap P(\tau)|}{|P(\tau)|},$$

where $|\cdot|$ is the set cardinality operator. Only proteins for which the prediction set is nonempty can be used to calculate average precision. To address this issue the root term is counted as a prediction for all proteins. Similarly, recall is defined as the fraction of experimental (true) terms which were correctly predicted, i.e.,

$$rc(T, P(\tau)) = \frac{|T \cap P(\tau)|}{|T|}.$$

As before, precision $pr(\tau)$ and recall $rc(\tau)$ for the entire data set are calculated as averages over the entire set of proteins (note that an alternative definition of precision and recall given by [13] is described in Sect. 2.1.8). Finally, to provide a single evaluation measure we use the maximum F-measure over all decision thresholds. For a particular set of terms T and $P(\tau)$, F-measure is calculated as the Harmonic mean of precision and recall. More formally, the final evaluation metric is calculated as

$$F_{\max} = \max_{\tau} \left\{ 2 \cdot \frac{pr(\tau) \cdot rc(\tau)}{pr(\tau) + rc(\tau)} \right\} \quad (2.12)$$

where $pr(\tau)$ and $rc(\tau)$ are calculated by averaging over the data set.

2.1.6.1 Information-Theoretic Weighted Formulation

The definition of information accretion and the use of a probabilistic framework defined by the Bayesian network enable the straightforward application of information accretion to weight each term in the ontology. Therefore, it is easy to generalize the definitions of precision and recall from the previous section into a weighted formulation. Here, weighted precision and weighted recall can be expressed as

$$wpr(T, P(\tau)) = \frac{\sum_{v \in T \cap P(\tau)} ia(v)}{\sum_{v \in P(\tau)} ia(v)}$$

and

$$wrc(T, P(\tau)) = \frac{\sum_{v \in T \cap P(\tau)} ia(v)}{\sum_{v \in T} ia(v)}.$$

Weighted precision $wpr(\tau)$ and recall $wrc(\tau)$ can then be calculated as weighted averages over the database of proteins, as in Eqs. 2.9 and 2.10.

In addition to weighted precision and recall our framework also facilitates calculating weighted specificity

$$wsp(T, P(\tau)) = \frac{\sum_{v \in T^c \cap P^c(\tau)} ia(v)}{\sum_{v \in T^c} ia(v)}$$

where T^c represents the complement of set T ($G \setminus T$).

2.1.7 Supplementary Evaluation Metrics

When calculating remaining uncertainty, misinformation, precision, and recall only consistent subgraphs of the Gene Ontology were considered. Under this framework, if a protein is annotated with multiple terms (either experimentally determined or predicted), as in Fig. 2.2, we determine consistent graphs T (true) or P (predicted) by recursively propagating annotations toward the root(s) of the ontology and taking a union of all terms encountered along the way. In each of these measures, it is sufficient to only consider vertices (terms) in the annotation graphs and calculate the similarity measure by manipulating vertices in an additive fashion. For example, each vertex in T or P counts equally in the precision/recall-based evaluation while the information accretion is used to weigh the vertices in the ru-mi-based evaluation.

A distinctly different approach can be taken by considering, on an individual basis, each leaf term that comprises a set T or P . This is the approach taken to calculate various information-theoretic metrics [6, 8, 9, 11, 12] as well as to provide an alternative definition of precision and recall [13]. In this context the sets of leaf terms that define T and P (which we refer to as $\mathcal{L}(T)$ and $\mathcal{L}(P)$ respectively, and formally introduce below) are used to calculate a given metric. After calculating all pairwise metrics between the leaf terms, several different methods for averaging these scores can be applied to create a single similarity (or distance) value between T and P . We discuss these approaches below.

2.1.7.1 Basic Definitions

Suppose we are given an ontology in the form of directed acyclic graph $G = (V, E)$, where V is a set of vertices and $E \subset V \times V$ is the set of edges. In this graph, given an edge $(u, v) \in E$, we refer to vertex u as a parent of v and, alternatively, to vertex v as a child of u . We also consider a set of all ancestors of v , $\mathcal{A}(v)$, and find this set by recursively identifying parents of all discovered nodes starting with v until the root(s) of the ontology is (are) reached. For mathematical convenience, we consider vertex v to be a member of $\mathcal{A}(v)$. Finally, given two vertices u and v , we define a set of common ancestor nodes between these two vertices as $\mathcal{A}(u, v)$. Thus, $\mathcal{A}(u, v) = \mathcal{A}(u) \cap \mathcal{A}(v)$.

Consider now a consistent annotation graph T , where the set of vertices in T is a subset of vertices in G . We define $\mathcal{L}(T)$, or the set of Leaf terms represented by T , as

$$\mathcal{L}(T) = \{u : u \in T \wedge \neg \exists ((u, v) \in E \wedge v \in T)\}. \quad (2.13)$$

In other words, $\mathcal{L}(T)$ contains only those vertices (terms) from T that do not have children in T . Thus, the leaf terms are defined with respect to a particular annotation graph T and generally differ from the leaf nodes in the ontology.

2.1.7.2 Information-Theoretic Metrics Between Pairs of Vertices

When calculating the information-theoretic metrics of [6, 8, 9, 11, 12], we calculate the information content of an individual term $v \in V$ as

$$i(v) = \log \frac{1}{\Pr(v)} \quad (2.14)$$

where $\Pr(v)$ can be calculated as the relative frequency of term v among experimentally annotated proteins. The semantic similarity between two distinct terms u and v as defined by [11] was calculated as

$$s_R(u, v) = \max_{w \in \mathcal{A}(u, v)} \{i(w)\}, \quad (2.15)$$

where $\mathcal{A}(u, v)$, as mentioned above, defines the set of common ancestors of terms u and v . Similarity as defined by Lin [8] was calculated as

$$s(u, v) = \frac{s_R(u, v)}{i(u) + i(v)}, \quad (2.16)$$

and as defined by Schlicker et al. [12] as

$$s(u, v) = \frac{s_R(u, v)}{i(u) + i(v)} \cdot (1 - \min_{w \in \mathcal{A}(u, v)} \{\Pr(w)\}). \quad (2.17)$$

Finally, the distance metric defined by [6] was calculated as

$$d(u, v) = i(u) + i(v) - 2 \cdot s_R(u, v). \quad (2.18)$$

2.1.7.3 Information-Theoretic Metrics Between Pairs of Graphs

Since the above metrics are only defined for two distinct terms, it is necessary to provide a mechanism to utilize these metrics in instances where a protein is annotated with graphs containing multiple leaf terms. Given two nonempty consistent annotation graphs of true and predicted terms, T and P , and the set of leaf terms that define each set, $\mathcal{L}(T)$ and $\mathcal{L}(P)$, we employed two strategies for averaging. In the first case, values were averaged between all possible pairs of terms in $\mathcal{L}(T)$ and $\mathcal{L}(P)$. Specifically, we calculated $s(T, P)$ as

$$s(T, P) = \frac{1}{|\mathcal{L}(T)| \cdot |\mathcal{L}(P)|} \sum_{t \in \mathcal{L}(T)} \sum_{p \in \mathcal{L}(P)} s(t, p). \quad (2.19)$$

We refer to this form of averaging as *all-pair* averaging. This method of averaging was applied by Lord et al. [9] in calculating similarity between two functional annotations.

In the second case we calculated the similarity between the two sets as the average of the maximum similarity between a term from one set and all terms in the other. Specifically, we calculated $s(T, P)$ as

$$s(T, P) = \frac{1}{2|\mathcal{L}(T)|} \sum_{t \in \mathcal{L}(T)} \max_{p \in \mathcal{L}(P)} \{s(t, p)\} + \frac{1}{2|\mathcal{L}(P)|} \sum_{p \in \mathcal{L}(P)} \max_{t \in \mathcal{L}(T)} \{s(t, p)\}. \quad (2.20)$$

This measure represents the technique of averaging employed by Verspoor et al. [13] when calculating precision and recall (originally referred to as hierarchical precision and recall). There, the authors separately calculate precision as

$$pr(T, P) = \frac{1}{|\mathcal{L}(P)|} \sum_{p \in \mathcal{L}(P)} \max_{t \in \mathcal{L}(T)} \frac{|\mathcal{A}(t, p)|}{|\mathcal{A}(p)|} \quad (2.21)$$

and recall as

$$rc(T, P) = \frac{1}{|\mathcal{L}(T)|} \sum_{t \in \mathcal{L}(T)} \max_{p \in \mathcal{L}(P)} \frac{|\mathcal{A}(t, p)|}{|\mathcal{A}(t)|}. \quad (2.22)$$

We refer to this method of averaging as *max-average*. Although not implemented here, Schlicker et al. [12] employ a technique for averaging that is similar to Eq. (2.20), but takes the maximum average similarity for one set as opposed to the average between the two. Specifically,

$$s(T, P) = \max \left\{ \frac{1}{|\mathcal{L}(T)|} \sum_{t \in \mathcal{L}(T)} \max_{p \in \mathcal{L}(P)} \{s(t, p)\}, \frac{1}{|\mathcal{L}(P)|} \sum_{p \in \mathcal{L}(P)} \max_{t \in \mathcal{L}(T)} \{s(t, p)\} \right\}. \quad (2.23)$$

When averaging pairwise comparisons for distance metrics, the above averages are calculated as the average of minimum pairwise distances instead of maximum pairwise similarities.

2.1.8 Additional Topological Metrics

In addition to information-theoretic metrics, we also used Jaccard's similarity coefficient [5] when calculating the similarity between the two consistent annotation graphs T and P . The Jaccard similarity coefficient is defined as

$$s(T, P) = \frac{|T \cap P|}{|T \cup P|}. \quad (2.24)$$

We note that cosine similarity as well as Maryland bridge coefficient [3] usually result in values correlated with the Jaccard similarity coefficient. For that reason, these two similarity measures were not presented.

2.2 Confusion Matrix Interpretation of ru and mi

While we introduce remaining uncertainty and misinformation in the context of comparing subgraphs of a larger ontology these two terms can also be intuitively interpreted as the information-theoretic analogs of false positives and false negatives or Type I and Type II errors. If we consider the confusion matrix in Table 2.2, we see the four positive outcomes that can occur when attempting to perform binary classification. False positives (Type I errors) are instances where a data point is a negative example but is incorrectly classified as a positive. In the context of hypothesis testing, these are cases where the Null hypothesis is true, but has incorrectly been rejected. False negatives (Type II errors) represent cases where the data point is a positive, but has incorrectly been labeled as a negative. These are cases where the Null hypothesis is false, but a model has failed to reject it.

Given a set of true terms, T , and predicted terms, P , then $P \setminus T$ would represent the terms that would fall in the false positive or Type I error portion of the confusion matrix. As defined in Eq. (2.6), this is the same set of terms whose joint probability, and subsequently information content, are measured when calculating misinformation. In this context, misinformation can be thought of as the number of bits of information, instead of traditional counts, the false positives represent. Conversely, if we consider false negatives, or the set of terms defined by $T \setminus P$ we are referring to

Table 2.2 A sample confusion matrix showing the four potential outcomes when performing binary classification

		True label	
		Positive	Negative
Predicted label	Positive	True positives	False positives (Type I)
	Negative	False negatives (Type II)	True negatives

the same set of terms used to calculate remaining uncertainty in Eq. (2.5). Remaining uncertainty, in this context, can be thought of as the number of bits of information, instead of traditional counts the false negatives represent.

2.3 Annotation Models

In order to judge the validity of our evaluation metrics, we implemented several well-studied annotation models. This was done by first collecting all proteins with GO annotations supported by experimental evidence codes (EXP, IDA, IPI, IMP, IGI, IEP, TAS, IC) from the January 2011 version of the Swiss-Prot database (29,699 proteins in MFO; 31,608 in BPO; and 30,486 in CCO). We then generated three simple function annotation models: Naïve, BLAST, and GOTcha, in order to assess the ability of performance metrics to accurately reflect the quality of a predicted set of annotations. In addition to these three methods, we generated another set of “predictions” by collecting experimental annotations for the same set of proteins from a database generated by the GO Consortium released at about the same time as our version of Swiss-Prot. This was done to quantify the variability of experimental annotation across different databases using the same set of metrics. In addition, this comparison can be used to estimate the empirical upper limit of prediction accuracy because the observed performance is limited by the noise in experimental data. All computational methods were evaluated using 10-fold cross-validation.

2.3.1 The Naïve Model

The Naïve model was designed to reflect biases in the distribution of terms in the data set and was the simplest annotation model we employed. It was generated by first calculating the relative frequency of each term in the training data set. This value was then used as the prediction score for every protein in the test set; thus, every protein in the test partition was assigned an identical set of predictions over all functional terms. The performance of the Naïve model reflects what one could expect when annotating a protein with no knowledge about that protein.

2.3.2 The BLAST Model

The BLAST model was generated using local sequence identity scores to annotate proteins. Given a target protein sequence x , a particular functional term v in the ontology, and a set of sequences $S_v = \{s_1, s_2, \dots\}$ annotated with term v , we determine the BLAST predictor score for function v as

$$\max_{s \in S_v} \{sid(x, s)\},$$

where $sid(x, s)$ is the maximum sequence identity returned by the BLAST package [1] when the two sequences are aligned. We chose this method to mimic the performance one would expect if they simply used BLAST to transfer annotations between similar sequences.

2.3.3 The GOTcha Model

The third method, GOTcha [10], was selected to incorporate not only sequence identity between protein sequences, but also the structure of the ontology (technically, BLAST also incorporates structure of the ontology but in a relatively trivial manner). Specifically, given a target protein x , a particular functional term v , and a set of sequences $S_v = \{s_1, s_2, \dots\}$ annotated with function v , one first determines the r-score for function v as

$$r_v = c - \sum_{s \in S_v} \log(e(x, s)),$$

where $e(x, s)$ represents the E-value of the alignment between the target sequence x and sequence s , and $c = 2$ is a constant added to the given quantity to ensure all scores were above 0. Given the r-score for function v , i-scores were then calculated by dividing the r-score of each function by the score for the root term $i_v = r_v / r_{\text{root}}$. As such, GOTcha is an inexpensive and robust predictor of function.

References

1. Altschul, S.F., et al.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**(17), 3389–3402 (1997)
2. Clark, W.T., Radivojac, P.: Information-theoretic evaluation of predicted ontological annotations. *Bioinformatics* **29**(13), i53–i61 (2013)
3. Glazko, G., et al.: The choice of optimal distance measure in genome-wide datasets. *Bioinformatics* **21**(Suppl 3), iii3–iii11 (2005)
4. Guzzi, P.H., et al.: Semantic similarity analysis of protein data: assessment with biological features and issues. *Briefings Bioinf.* **13**(5), 569–585 (2012)

5. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull. Soc. Vaud. des Sci. Nat.* **37**, 574–579 (1901)
6. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: *International Conference on Research in Computational Linguistics*, pp. 19–33 (1997)
7. Koller, D., Friedman, N.: *Probabilistic Graphical Models*. The MIT Press, Cambridge (2009)
8. Lin, D.: An information-theoretic definition of similarity. In: *Proceedings of the 15th International Conference on Machine Learning*, pp. 296–304. Morgan Kaufmann, San Francisco (1998)
9. Lord, P.W., et al.: Investigating semantic similarity measures across the gene ontology: the relationship between sequence and annotation. *Bioinformatics* **19**(10), 1275–1283 (2003)
10. Martin, D.M., et al.: GOTcha: a new method for prediction of protein function assessed by the annotation of seven genomes. *BMC Bioinf.* **5**, 178 (2004)
11. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448–453 (1995)
12. Schlicker, A., et al.: A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinf.* **7**, 302 (2006)
13. Verspoor, K., et al.: A categorization approach to automated ontological function annotation. *Protein Sci.* **15**(6), 1544–1549 (2006)

Information-Theoretic Evaluation for Computational
Biomedical Ontologies

Clark, W.T.

2014, VII, 46 p. 12 illus., 6 illus. in color., Softcover

ISBN: 978-3-319-04137-7