# Starfish

Connecting the Docs

Finding implicitly related items based on semantic similarities and metadata in a non-hierarchical network of documents

**Authors**

R. van Ginkel | J. Peters | L. Weerts

Project commissioned by *Perceptum B.V.*

**Supervisors**

| | |
|---|---|
| Academic Supervisor | Raquel Fernandez |
| Company Supervisors | Robrecht Jurriaans |
| | Sander Latour |
| | Wijnand Baretta |

June 27, 2014

Universiteit van Amsterdam

**Abstract**

This report describes the results of the second year's project for the Perceptum team. The project focused on creating a *document link recommender system* to the document knowledge base of the Starfish website, a platform where educators can share information about educational innovation. Because users of Starfish do not have knowledge of all the documents in the entire knowledge base, a system that can perform this automatically is needed. This report describes the implementation and analysis of several algorithms that can perform this task.

To create the document linker, a document descriptor-based technique was used. Firstly, each of the documents is transformed into a descriptor by algorithms called *vectorizers*. Six vectorizers were implemented. Two vectorizers are based on the text of documents (textvectorizer and weighted text vectorizers) and two others use the tags of documents (simple tag similarity and tag smoothing). The last two vectorizers perform the textual transformation on text-based descriptions of tags (glossaries of tags and weighted tags). Additionally, a hybrid method of a text-based and tag-based is proposed. After creating the document descriptors, a ranking is made of all documents based on the similarity of the document descriptors and the descriptor of the new added document. This was done using the K-Nearest Neighbor algorithm with several distance metrics, of which the cosine distance turned out to work the best. The system also provides a method that can re-rank this set of proposed documents based on the probability that two documents are linked together. The last step in the pipeline determines how many of the nearest neighbours should be returned. For this a threshold was set that compares the distance as calculated with K-Nearest Neighbor of two documents with consecutive ranks.

Three main conclusions were drawn from this study. Firstly, the text vectorizers performs the best if the newly added document is a Question (42.02-44.82% accuracy on k-link measuring). However, it cannot deal with documents that have different languages or non-textual documents such as images, videos and audio. The simple tag vectorizer has the best performance (22.80% overall average accuracy on k-link measuring) and is the fastest. The best overall performance with the k-link measurement is gained with the hybrid vectorizer (26.13%) that uses the textvectorizer if no tags are available and the simple tag vectorizer otherwise. This vectorizer performs as good on most document types as the simple tag vectorizer, but performs significantly better on questions (31.93% versus 16.67%). Secondly the probabilistic model of the network that is proposed is either to simplistic or the data available is too little. In either case it might be off interest to further investigate a similar model on a bigger data set. Lastly the method of selecting the number of documents shows that the overall performance does not change significantly if the threshold is added to most vectorizers. However, the text vectorizers seem to have a bias towards a higher precision in the trade off between precision and recall. The glossaries of tags and weighted glossaries of tags get a higher recall for persons. The best performance while using the threshold was obtained using the hybrid vectorizer, with an average precision, recall and F1 measure of respectively 28.61%, 27.26% and 23.92%.

These results are clearly not good enough for an automatic linking system, but could be considered high enough for a recommender system since they are far above guessing level. It is now up to the client to choose if a precision of 28.61% is good enough to let the user select a document to link and if a recall of 27.26% covers enough of the documents within the knowledge base.

# Contents

# 1 Introduction

This report describes the results of the second year's project for the Perceptum team. The project focused on creating a *document link recommender system* to the Starfish website.

Starfish, one of Perceptum's products, is a website that aims to share knowledge about the education domain by means of a connected graph. People from all around the world should get access to this knowledge graph in a simple, personalized manner. The nodes in this graph are documents and they are connected with links. These documents can be of all sorts of types - e.g. a good practice, information or a question. Each document has a set of tags associated with it, which describe the different aspects of educational innovation it covers. Starfish is community-driven: both the content of the documents as the links between documents are determined by the users of Starfish. The drawback of a community-driven knowledge graph is that not all the users have complete knowledge the entire document base. Therefore, many links will not be made because the users are unaware of the documents that they could link to. A possible solution could be to make use of administrators, who can devote more time to get to know all the documents. However, that approach has two drawbacks. First of all, this would mean that some central authority determines whether or not two documents should be linked, something which strictly opposes the idea of a community-driven knowledge base. Secondly, if the knowledge base grows even larger, it becomes impossible for an administrator to keep track of all documents. Imagine one person having to link all pages on Wikipedia - an impossible job.

In order to overcome the problem of linking documents in a large knowledge base, this process should be automated. This project focuses on the automatization of connecting documents within Starfish. Though ideally these connections should be made completely automatic, a first step would be to create a recommendation system. When a user adds a new document, he or she can choose from a list of proposed documents the documents he or she deems relevant. This means that the recommender system does not have to work perfectly, but should work well enough. Defining 'well enough', however, is also a part of this project. Thus, the product vision of the system can be described by the following:

**Product vision:**

**For** StartFish users **who** search for and edit knowledge in Starfish, **the** document linker **is a** core system addition to Starfish **that** finds related documents. **Unlike** linking by moderation (centralized) our product uses algorithms and data (decentralized) to automatically suggest document links.

Within the time span of this project multiple ways of recommending links between documents have been explored. The results of these explorations will be discussed in this report.

# 2 Domain

The Starfish website aims to be a platform for educators where educational innovations and projects can be shared. Because many teachers have different vocabulary and diverse questions, a strict hierarchical structure of the shared content is difficult to achieve. Starfish tries to overcome this by structuring it's knowledge base in a non-hierarchical manner. There are sub communities within Starfish which gives learners the opportunity to share knowledge that is specific to their faculty or institution.

The knowledge base itself consists of one big set of entities. These entities are called documents. Currently, the Starfish knowledge graph contains 240 documents. These documents can be of a variety of types. Each document in this graph is of one of the following types: Information, Glossary, Question, Good Practice, Project, Person or Event. Documents have an 'author'-field, 'title'-field, 'text'-field. The Person-type is an exception, since has a 'name'-field instead of 'title', an 'about'-field instead of 'text' and no author. Each document also has a set of tags and links assigned to it. Some document types have different optional fields like 'headline' for good practices and projects. The graph is structured by directional links between documents. On average a document has 3.9 links.

Each document in the knowledge base is assigned a set of tags based on the different aspects of educational innovation the document covers. On average a document has 3.4 tags. Glossaries are special types of documents, as they are description for tags. This means that it is unnecessary for the link recommendation system to return glossaries, since a glossary is 'linked' via an assigned tag. Because different groups use alternative names to describe concepts, tags can be aliases of each other. For this project, aliased tags are regarded as one single tag. Of the 210 tags the current system contains, 146 unique tag concepts can be distinguished of which 24.6% have a glossary.

These numbers and properties of the system give some insight into the current state of the dataset and possible solutions. The major part of Starfish data is text-based, so semantical document analysis could be performed with standard text processing techniques. The tags of documents could also give insight into the semantics of the documents. Additionally, the links that are currently in Starfish can be used as guidelines on when documents should be linked and when not.

Although Starfish aims to be user-driven, user voting will not be explored in this project. This was not part of the initial request and Starfish is currently to small for such an endeavor. The data for this is not at hand and there are too little users to test such a system effectively.

# 3 Theory

In solving the linking problem, one must find a way to compare different documents based on their linkability with newly added documents. Computationally, this can be done by creating document descriptors – vectors that describe the features of a document in some way. Given the Starfish domain, these descriptors can be created using a tag-based or text-based approach. This section will elaborate the background of the descriptor-based approach.

## 3.1 Text-based descriptors: bag of words and TF-IDF

One way of capturing the semantic similarity of two text document is by comparing the TF-IDF values of their contents. If two documents cover the same subject(s), they are likely to contain similar keywords. To capture this similarity, the documents can be transformed into a list of all words that are present within that text. This is called a bag-of-words representation. Instead of counting the frequency of each word within a document, the more sophisticated Term Frequency-Inversed Document Frequency value can be used. TF-IDF is a statistic that reflects the importance of a word in a document within a corpus and can be calculated as follows:

$$\mathrm{tf}(t, d) = 0.5 + \frac{0.5 \times f(t, d)}{\max\{f(w, d) : w \in d\}} \tag{1}$$

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

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

The TF-IDF induces a trade off between the *Term Frequency*, the number of times a word appears in a document, and the *Inverse Document Frequency*, the inverse of how often a word is used in the entire corpus. Words such as 'and', 'or', and 'of' will have a high term frequency within a document. However, their inverse document frequency will be very low, since they occur very often within the entire corpus. Infrequent used words such as 'Starfish' are less likely to occur within a corpus, so if they do occur often within one particular document the TF-IDF value will be high.

## 3.2 K-Nearest Neighbor

After creating the document descriptors as described in the previous section all documents lay in a specific vector space. Using different distance measures, the distances between the documents in the knowledge base and some new document can be computed. The $K$-Nearest Neighbor algorithm ranks the documents on their distance and selects the $K$ document descriptors that have the lowest distance. In other words, it selects the $K$ vectors that are most similar in the given vector space. See algorithm 1 for pseudocode of the $K$-Nearest Neighbor algorithm.

**Algorithm 1** $K$-Nearest Neighbor

```
docs = Vector document descriptors.
dist = Vector document distance metric.
d = Document to find neighbors of.
δ = []
for doc ∈ docs do
  δ ⇐ (doc, dist(d, doc))          // Compute distance with every document
end for
sorted = sort(delta)               // Sort from lowest to highest distance
return sorted[0...K]               // Return K first documents
```

### 3.2.1 Distance Metric

As discussed before, the $K$-Nearest Neighbor algorithm uses a distance metric to compute the distance between two document descriptors in a given vector space. A distance metric or simply called the distance is a function $d$ that maps two documents descriptors $X$ and $Y$ to the real numbers (4).

$$d : X \times Y \mapsto \mathbb{R} \tag{4}$$

This function $d$ at least satisfies the following conditions for the documents $X$, $Y$ and $Z$.

1. $d(X, Y) \geqslant 0$
2. $d(X, Y) = 0$ if and only if $X = Y$
3. $d(X, Y) = d(Y, X)$
4. $d(X, Z) \leqslant d(X, Y) + d(Y, Z)$

A set of distance metrics is discussed in the following paragraphs.

**Euclidean distance** The euclidean distance metric is a generalization of the pythagorean metric to higher dimensions. The euclidean distance is the ruler distance, which means that it is equal to the distance between the two heads of each vector. It is calculated as follows:

$$\begin{aligned} d_{\text{euclid}}(q, p) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \\ &= \sqrt{(q - p) \cdot (q - p)} \\ &= \|q - p\| \end{aligned}$$

This shows that the euclidean distance is the norm of the vector $p - q$ or $q - p$ (as $d(q, p) = d(p, q)$). This makes sense as the euclidean distance is the distance between the two vectors. An illustration of this is given in figure 1

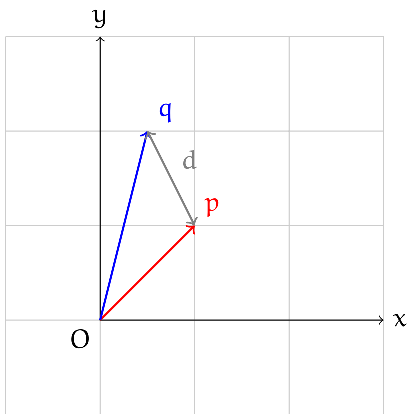


Figure 1: The euclidean distance $d$ between two document descriptors $p$ and $q$. The line marked $d$ is the euclidean distance between the vectors $p$ and $q$. It is easy to see that this is the 'ruler' distance between the heads of the two vectors.

**Cosine distance** The cosine distance measures the cosine of the angle $\varphi$ between two document descriptors. The intuition of this metric is that a document 'word$_1$ word$_2$' and 'word$_1$ word$_1$

word$_2$ word$_2$' probably are very similar as only their word frequencies differ. However, the euclidean distance computes a distance $> 0$. The cosine distance can recognise such a similarity by computing the angle between two documents. This is shown visually in figure 2. In this example both descriptors would point in the exact same direction, but the second descriptor is longer. This results in a cosine of zero, equal to an exact match. Though the cosine similarity theoretically could be negative, the values in the vectors will never take a value $< 0$ because a word can not occur less then zero times. Because of this the values of the cosine of the angle between two descriptors will range between 0 and 1, where 0 means no match at all and 1 is an exact match. Therefore the cosine distance can be defined as follows:

$$d_{\cos}(q, p) = |\cos(\varphi) - 1|$$

where $\varphi$ is the angle between $q$ and $p$. Given that $q \cdot p = ||q|| \, ||p|| \cos(\varphi)$, we can easily compute the $\cos(\varphi)$. This results in the definition of the cosine distance metric in equation 5. The cosine distance can be computed very efficiently on sparse vectors as only the non-zero values are important for the computation. Another advantage of the cosine similarity is that it is independent of document length. The cosine similarity does not strictly satisfy the conditions for a distance metric, but acts as if the vectors are normalized.

$$d_{\cos}(q, p) = \left| \frac{q \cdot p}{||q|| \, ||p||} - 1 \right| \tag{5}$$

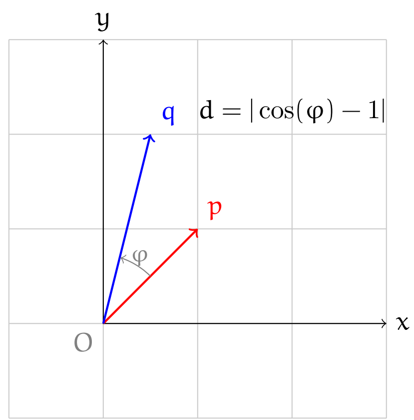


Figure 2: The cosine distance $d$ between two document descriptors $p$ and $q$. The cosine distance is defined by the cosine of the angle between the two vectors $p$ and $q$. Is is easy to see that the angle $\varphi$ is the angle between the two documents that defines the distance $d$.

**Bhattacharyya distance** The feature vectors used to represent documents can also be seen as unordered histograms. A way to compare histograms is the Bhattacharyya distance metric. The Bhattacharyya metric, as explained in Chaudhuri (2011), is used to compute the similarity of two discrete or continuous probability distributions. The metric ranges from 0 tot 1 where 0 is a perfect match. The Bhattacharyya distance metric is defined as:

$$d_B(q, p) = \sqrt{1 - \frac{1}{\sqrt{\overline{q}\,\overline{p}n^2}} \sum_{i=1}^{n} \sqrt{q_i p_i}} \tag{6}$$

where $\overline{q} = \frac{1}{n} \sum_{j=1}^{n} q_j$ and $n$ the number of elements in each of the vectors.

**Intersection** Another histogram similarity metric is the histogram intersection. The histogram intersection (Swain and Ballard, 1991) was originally introduced in the computer vision field and is defined in as shown in equation 7

$$d_{\text{int}}(q, p) = \sum_{i=1}^{n} \min(q_i, p_i) \tag{7}$$

In the computer vision domain this results in the number of pixels that have the same color. For the document descriptors used in this case this comes down to the weights the two documents have in common. In other words when two documents that have high weights for the same words the total will be higher then when two documents have hight weights for different words. This measure does not satisfy the four conditions for a distance metric. However, it may still be useful in the document similarity domain.

**Sample Correlation Coefficient** Finally, a method that is often used to compute the similarity between histograms is the correlation. Correlation measures the dependence between two sample sets. Pearson's correlation, when applied to a sample, is defined as follows:

$$d_{\text{corr}}(q, p) = \frac{\sum_{i=1}^{n}(q_i - \overline{q})(p_i - \overline{p})}{\sqrt{\sum_{i=1}^{n}(q_i - \overline{q})^2}\sqrt{\sum i = 1^n (p_i - \overline{p})}} \tag{8}$$

$$= \frac{(q - \overline{q}) \cdot (p - \overline{p})}{||q - \overline{q}||\,||p - \overline{p}||} \tag{9}$$

$$= d_{\cos}(q - \overline{q}, p - \overline{p}) \tag{10}$$

As shown above the Pearson correlation is equal to the cosine distance when the documents descriptors are first converted to have mean zero. This means that the Pearson correlation is, like the cosine distance, scale invariant. However, unlike the cosine distance the Pearson correlation is also shift invariant. This means that when a scalar is added to all elements of the documents descriptor the correlation is still the same.

$$d_{\text{cor}}(p, q) = d_{\text{cor}}(p, q \times 2 + 3)$$

# 4 Product overview

The product created in this project is a python program takes a set of documents and a new document and returns the subset of documents that should be linked with the new document. For this, a descriptor-based approach was used, which consists of three steps. First, each of the documents is transformed into a descriptor. Second, a ranking is made of all documents based on the similarity of the document descriptors and the descriptor of the new added document. This was done using the K-Nearest Neighbor algorithm with several distance metrics. Lastly, an algorithm chooses the proper amount of proposed links that must be returned. We will now discuss each of these steps, since these will give more insight into the approach that was chosen to solve the problem. For the performance of the different algorithms we refer to the experiment section.

## 4.1 Vectorizer

The first step is to create document descriptors, which is done by algorithms that called *vectorizers*. Two main paths have been explored: transformation based on text and transformation based on tags.

### 4.1.1 Text-based transformation

**Textvectorizer** The text-based vectorizers use the textual content of the documents and are therefore generally applicable to knowledge bases that contain text-based documents. The textual content is first transformed into a bag of words. Then, based on all the documents in the knowledge base, the *TF-IDF* value is calculated for each of the words in the bag of words.

**Weighted_textvectorizer** The weighted textvectorizer is implemented as an extension of the textvectorizer. First, all descriptors of the documents are calculated similarly as in the textvectorizer. Then each document descriptor is recursively increased with the sum of the descriptors of it's links, decreased by some weight parameter. This captures the idea that if a new document resembles some of the documents that are linked to one particular document, it is more likely to be linked to this particular document.

### 4.1.2 Tag-based transformations

**Simple_tag_similarity** The tag-based transformations are more Starfish specific, since these make use of the tags that are assigned to the documents - Starfish feature. A tag is a keyword that describes a topic/term that is important for that document. For example, 'Online Support and Online Assessment for Teaching and Learning Chemistry' is tagged with 'chemistry', 'e-learning' and 'assessment'. The simple tag similarity vectorizer creates a vector where each value indicates whether or not one particular tag is assigned to the document.

**Tag_smoothing** The tag smoothing vectorizer uses the co-occurence of tags in estimating document similarity. Even though tags might not co-occur on any document in the data set, they can still provide information about each other. For example, the dataset consists of documents with associated tags like $\{\{\mathtt{t}_1, \mathtt{t}_2\}, \{\mathtt{t}_1, \mathtt{t}_3\}\}$. From the co-occurence it does not follow that $\mathtt{t}_2$ and $\mathtt{t}_3$ are related, however by transitivity with $\mathtt{t}_1$ we want to create a small implicit link between $\mathtt{t}_2$ and $\mathtt{t}_3$. The tag smoothing method does this based on work from Zhou et al. (2011).

**Glossaries_of_tags** Another way of capturing tag similarity is by using tag Glossaries. They can be used by applying a text-based transformation on the glossaries to find similarities between tags. Thus, glossaries_of_tags can be seen as a hybrid form of the tag and text-based approaches, where the glossary of a tag is turned into a TF-IDF bag of words. The document descriptor consists of the sum of vectors of each of its tags.

**Weighted_tag_vectorizer** This is an extension of glossaries of tags. In the original glossaries of tags, it is assumed all tags contribute the same amount of information to a document's links. In practice some tags provide more information than others. If a certain tag is on nearly all documents in the dataset, it does not provide a lot of insight into linking new documents. In contrast a tag which is only attached to a small subset of documents is much more informative. The weighted tag vectorizer creates descriptors by summing the tag vectors with a weight based on the frequency of that tag in the dataset. The intuition for this is the same as that of the TF-IDF bag of words approach for creating vector representations of a text.

**Hybrid** The hybrid vectorizer is a combination of the text and simple tag vectorizers. If a document does not have tags, the textvectorizer is used. Otherwise, the simple tag vectorizer is used to propose document links.

## 4.2 Distance

A ranking is created using the K-Nearest Neighbor algorithm that sorts the document descriptors based on their distances with the newly added document. The following five distance metrics were implemented. These are discussed in section 3.2.1

- Eucledian
- Cosine
- Bhattacharyya
- Correlation
- Intersection

The cosine distance is the default value, since that one seems to perform the best on the Starfish knowledge graph based on our experiments. More on this in section 6.

## 4.3 Starfish specific adaptations: Bayesian weighting

Both the tag-based and text-based approaches use some kind of 'semantic similarity' - the similarity of tags or text. However, except for the weighted text vectorizers, no information about possible links is used. For example, the text on a person's profile might be similar to other persons, but within Starfish a Person document is almost never linked to another Person. To make more use of known links within the Starfish knowledge base, the ranking of document descriptors as created by K-Nearest Neighbor can be re-ranked using the probability that two documents are linked together given their tags:

$$P(D_a \rightarrow D_b | t)$$

Thus, the weight of a tag within a vector is equal to the chance that given this particular vector, a document of type a (the type of the newly added document) and a document of type b (equal to the type of proposed link) are linked. The inverse of an approximation of this probability is multiplied with the distances that come from K-Nearest Neighbor in order to enlarge the distance of proposed links that are unlikely given the Starfish knowledge base.

## 4.4 Threshold value

The next step in the pipeline is to determine how many of the nearest neighbours should be returned. Depending on the application of the Starfish document linker, the desired number might vary. If one wants to immediately link the results, the certainty for relatedness should be high. If the links are presented to a user which can approve or reject them, the relatedness may be lower. Currently, this is configurable by setting a threshold parameter between 0 and 1. Zero will only return the closest document, 1 will return almost all. After exploration of the dataset the default value is 0.3, which roughly returns the same amount of links which is currently average in Starfish.

## 4.5 Output

The document linker can be run using the documentlinker.py file. There are two ways in which the results of the document linker are reported: a JSON file with the proposed links and a textual performance report. The JSON file can be viewed using *view.html*, a HTML page that can be opened in a browser. A file can be selected using the 'choose file'-button. The HTML page then displays a list of all documents. The content and proposed links can be viewed by clicking on the corresponding buttons. The grey links indicate False Negatives, the green ones True Positives and the red ones False Positives.

The performance reports is displayed in the terminal and shows the precision and accuracy (see the metric section in experiments) of the entire knowledge base and per document type. It also gives insight into the distribution of document types by presenting the percentages one type linked to another. For example, a percentage of 85% on the Person row and Question column indicates that of all Person documents, 85% of the links from a Person to another document types directed to Questions.

# 5 Method

## 5.1 Data preprocessing

Starfish text content is serialized as HTML, a format which is not suitable for calculating semantical similarity. Before processing, the data is sanitized by removing HTML tags and entities and convert unicode characters to their closest ASCII representation.

The raw starfish data contains tags which are aliases of other tags. During preprocessing all tags are replaced by their alias which have a glossary. Also, the current graph also contains some links which should not be returned by a link recommender system. Links from article to an author should be ignored as an author is automatically linked from the document. Links to glossaries should never occur, as they are linked through tags if appropriate.

After the data is sanitized, the tags are unique and the inappropriate links are removed, the data is ready for processing.

## 5.2 Text descriptors

### 5.2.1 Textvectorizer

The first set of vectorizers focuses on the texts of the documents. The *textvectorizer* is a very generic approach that can be used on any corpus of textual documents. In the Starfish context, the content of a document is defined as the 'title' and 'text'-fields of a document. The only exception on this are Persons, of which the 'name' and 'about'-fields are used.

The textvectorizer first transforms the set of documents into a bag of words and calculates the TF-IDF values for all words. This was done using the TFIDFVectorizer of scikitlearn (Pedregosa et al., 2011). Though the TF-IDF values of words that are used very often should be low, common words such as ‘and’, ‘or’ and ‘of’ are still present in the vectors. This could be caused by the different types of documents. For example, a Question often structured in a less complex way than a Project description. To prevent this from happening, the English stopwords list that comes standard with scikit learn was used to remove these words from the document descriptors.

### 5.2.2 Weighted textvectorizer

The *weighted text vectorizer* is an extension of the textvectorizer that takes into account the links of the proposed documents. The vectors of the links of a document are added with some weight to the vectors of the documents themselves. Intuitively, this would add semantic information about a document based on its links. For example, a Person is likely to write other documents about his or her subjects of expertise. Knowing not only the biography of a Person, but also the content he or she has added to Starfish, gives a more complete image of what documents could be related to that Person.

The vectors of links of a document are added in a recursive way, where documents that are linked directly have a higher weight than documents that are linked transitively. The recursive algorithm is displayed in algorithm 2.

**Algorithm 2** Weighted textvectorizer

```
Function weightedTextVectorize(document, depth, weight)
descriptor = TF-IDF(document)                        // Convert document to vector
if depth == 0 then
  return descriptor
end if
linkdescriptor = [0, 0, ..., 0]
for link ∈ links(document) do
  linkdescriptor += weight × weightedTextVectorize(link, depth-1)
end for
linkdescriptor = linkdescriptor / |links(document)|  // Divide by the number of links
return descriptor + linkdescriptor
EndFunction
```

## 5.3 Tag descriptors

### 5.3.1 Simple tag vectorizer

The tag-based approach is more Starfish specific than the text-based approach, since it depends on the tags that are available in Starfish. The tags on Starfish are added by the users themselves, so offer a human-based vision on what subject(s) a document really covers. The *simple tag vectorizer* is a very straightforward implementation of the idea of using tags. The vectors of this transformation consist of a binary list that tells whether or not a tag is attached to the document.

### 5.3.2 Tag smoothing

The *tag smoothing vectorizer* creates descriptors based on the tag set of a document. If a tag co-occurs with other tags in a document, we assume that these tags are similar. In this approach it is assumed that documents with such similar tags should be linked in Starfish. Let the frequency of occurrence with other tags across the dataset form a vector for each tag. The descriptor for a document is then created by combining the occurrence vectors for all the document's tags. Now documents with tags that occur together will be seen as similar.

There are two reasons why one would like to smooth the tag co-occurences. Firstly, a problem for this is that tags must occur together before the algorithms works properly. The Starfish dataset contains a lot of tags that only occur with a small frequency, which means the tag occurrence vector will contain many zeros. This makes the algorithm perform bad with little data. Secondly, two tags can describe the same concept and be connected to that concept through a common co-occurence with another tag. Whilst they describe the same concept and are connected to that, they are not directly linked together. Therefore it seems feasible to perform some sort of smoothing on the co-occurences of tags.

Zhou et al. (2011) proposed a method to cluster web documents based on tag set similarity. This is based on a similarity between two tags as a relation between the frequency these tags occur separate and together, as described in equation 11. To smooth these similarities between tags, a tag similarity matrix $\mathcal{C}$ is constructed. Each entry $c_{i,j}$ in this matrix can be viewed as the angle $\theta_{i,j}$ between two unknown vectors $\nu_i$ and $\nu_j$. These vectors cover both explicit similarity and implicit similarity (Park et al., 2010). This transfers the problem to find a set of linearly independent vectors $\{\nu_1, \nu_2, \ldots, \nu_n\}$ for which for all $\nu_i \cdot \nu_j = \cos(\theta_{i,j})$. One must find a matrix $\mathcal{V}$ for which $V^T V = C$. This can be done by orthogonal triangularization on $\mathcal{C}$ for which Zhou et al. introduces a modified Cholesky transform.

$$s_{i,j} = \frac{f_{i,j}}{f_i + f_j - f_{i,j}} \tag{11}$$

### 5.3.3 Glossaries of tags

The *glossaries of tags* approach is also based on the intuition that certain tags cover overlapping concepts. Just like the simple tag vectorizer the glossaries of tags approach exploits this intuition. In the Starfish system a tag is expected to have a glossary; a short English description of the concept of a tag. These glossaries contain terms and words that are descriptive for the tag. The glossary of tags vectorizer aims to use these terms by creating a document descriptor based on the descriptions of the tags that are associated with a document.

First for each tag a tag descriptor $t_i$ is created. This tag descriptor is a TF-IDF bag of words vector created from the glossaries of each tag. These descriptors form the basis for the document descriptors. For each document a document descriptor $d_i$ is created by summing the tag descriptors of the associated tags:

$$d_i = \sum_{j \in T(d_i)} t_j \tag{12}$$

where $T(d_i)$ is the set of tag indices associated with document $d_i$. This way a document descriptor is created based on a semantic representation of the tags associated with the document.

### 5.3.4 Weighted tag vectorizer

The *weighted tag vectorizer* is similar to the glossaries of tags vectorizer discussed in the previous section. However instead of just summing the tag descriptors an weight is used to scale the tag descriptor. This way tags that are associated with a lot of documents don't contribute as much to the document descriptor as tags that are only associated with a few documents. The weight

$w_t$ for each tag descriptor $t$ is based on the number of documents that the tag is associated with. $w_t$ is defined as follows

$$w_{t_i} = 1 - \frac{\text{Number of documents associated with } t_i}{\text{Total number of documents}} \tag{13}$$

With this weight the weighted tag document descriptor $d_i$ can be computed in the following way:

$$d_i = \sum_{j \in T(d_i)} w_{t_j} t_j \tag{14}$$

This vectorizer first gives extra importance to descriptive words in each of the tag descriptors by using the TF-IDF bag of words. After this more importance is given to tags that are more descriptive using the weights. This results in a document descriptor that has high weights for words that are descriptive for tags which themselves are descriptive for the document.

### 5.3.5 Hybrid

The *hybrid* vectorizer is a hybrid of the textvectorizer and simple tag similarity vectorizer. If a document has no tags, this vectorizer uses the textvectorizer. Otherwise, the simple tag vectorizer is used. This prevents the simple tag vectorizer from randomly assigning documents with one tag to a document that has no tags.

## 5.4 Bayesian weighting

Up to this point only the semantic similarity of documents has been taken into account. This does however, not take the information in the current network into account. For example, the probability of a document of type information linking to a document of type information is higher than that of the same document linking to a document of type project. For this $P(d_i \to d_j \mid \Sigma_{ij})$ and $P(d_i \to d_j)$ are interesting, which are the probability that two documents are linked given the number of tags they have in common and the probability that two document of a given type are linked, respectively. These probabilies can be estimated from the data. First it is shown how to compute $P(d_i \to d_j \mid \Sigma_{ij})$.

$$P(d_i \to d_j \mid \Sigma_{ij}) = \frac{P(\Sigma_{ij} \mid d_i \to d_j)P(d_i \to d_j)}{P(\Sigma_{ij})} \tag{15}$$

$$P(\Sigma) = \frac{\binom{D_\Sigma}{2}}{\binom{|D|}{2}} \tag{16}$$

$$P(\Sigma \mid d_i \to d_j) = \frac{\text{Number of links that have } \Sigma \text{ tags in common}}{|L|} \tag{17}$$

$$P(d_i \to d_j) = \frac{|L|}{\binom{|D|}{2}} \tag{18}$$

Where $L$ is the set of all links and $D$ is the set of all documents. $\Sigma_{ij}$ is the number of tags that document $d_i$ and $d_j$ have in common. As is shown above $P(d_i \to d_j)$ is computed as a uniform probability for all documents. However, this can also be computed while taking the document types into account. This results in a probability for every document type to every document type. In total this are $\tau^2$ probabilities where $\tau$ is the number of document types.

$$P(d_i \to d_j) = \frac{|\{(m, n) \mid m \to n \in L \wedge \tau(m) = \tau(d_i) \wedge \tau(n) = \tau(d_j)\}|}{|L|} \tag{19}$$

Where $\tau(x)$ is the type of document $x$. In other words, the number of links that have the type of $d_i$ as from document and the type of $d_j$ as to document divided by the total number of links. The results of this for the current network are displayed in table 1. This clearly shows that a link from a type have preference to other types. This information should be usefull when filtering bad links. These probabilities will be used to scale the distances that are computed by the nearest neighbor algorithm. Because the distances for documents that have a high probability should be low therefore the final distances are computed in the following way:

$$d_i = (1 - P(d_i \to d_j \mid \Sigma_{ij})) \times (1 - P(d_i \to d_j)) \times \delta_i \tag{20}$$

where $\delta_i$ is the distance as computed by the nearest neighbor algorithm and $d_i$ the final distance that is used to select the proposed documents to be linked. This may seem a rather simple statistical model. However, due to the limited set of items in the dataset it is very hard to learn a more complicated model.

| | Informat | Glossary | Question | Good Pr. | Project | Person | Event |
|---|---|---|---|---|---|---|---|
| Information | 0.069544 | 0.057554 | 0.015588 | 0.008393 | 0.010791 | 0.065947 | 0.008393 |
| Glossary | 0.094724 | 0.052758 | 0.008393 | 0.011990 | 0.021583 | 0.070743 | 0.004796 |
| Question | 0.017986 | 0.013189 | 0.022782 | 0.011990 | 0.011990 | 0.021583 | 0.005995 |
| Good Practice | 0.009592 | 0.008393 | 0.007194 | 0.003597 | 0.004796 | 0.009592 | 0.005995 |
| Project | 0.014388 | 0.016787 | 0.007194 | 0.003597 | 0.011990 | 0.017986 | 0.007194 |
| Person | 0.092326 | 0.074341 | 0.017986 | 0.009592 | 0.019185 | 0.002398 | 0.011990 |
| Event | 0.008393 | 0.004796 | 0.001199 | 0.004796 | 0.003597 | 0.009592 | 0.00479 |

Table 1: The probability that a document of a type in the left column links to a document of a type in the top row as calculated using equation 19.

## 5.5 Thresholds

After creating the descriptors for a document and finding a new document's nearest neighbors, a subpart of those links needs to be returned by the linker application. An average document in Starfish has 3.9 links. To create a dynamic threshold which returns similar a similar amount of documents, a document from the Starfish data is extracted and then to re-inserted to test the number of links the system returns to the number of links the document had in the original system. Figure 3 shows the distribution of outgoing links in the current network.

The threshold should depend on multiple factors which mostly depend on the setting in which the document linker will be used. Two major factors determine how the threshold should work.

1. The degree of certainty we expect from the returned documents.

2. The maximum number of documents that should be returned for a specific application.

If the system is used to directly create the links into the Starfish system, the first aspect is very important. Only documents with a very high degree of linking certainty should then be returned. The amount should be based on the contents of the current system. If the system is used to create recommendations which a user must accept or reject, the first aspect becomes less important and the system should return an amount of links which can quickly be reviewed by users. To ensure the flexibility for choice of algorithm and integration of the application, the threshold algorithm will contain a configurable parameter.

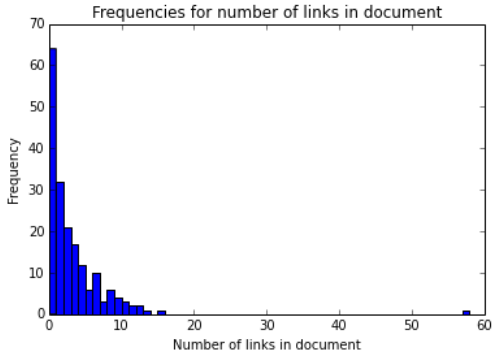


Figure 3: The frequency of outgoing links in Starfish documents

When adding a document to Starfish, it is assumed that there is always at least one related document. Based on the distance of the nearest neighbor for the new document, the index of a document and the configurable parameter, the threshold, is defined as in equation 21. In this equation, $\alpha$ is the configurable parameter, $m$ is the number of documents returned by the nearest neighbor algorithm, $d_0$ is the distance to the closest document, $\frac{m-n}{m}$ is a factor that ensures the maximum allowed distance decreases for documents ranked further away. This is based on the differences between two nearest neighbors which are visualized in figure 7. The distances have a long tail form: the distances between the nearest neighbors is relatively large for the closest documents and smaller between the documents further away.

$$t_n = \alpha(1 - d_0)\frac{m - n}{m} \tag{21}$$

# 6 Experiments

## 6.1 Evaluation metrics

In order to evaluate the performance of the different algorithms, two different metrics were used. Both methods use a 'take one out principle', where a document is taken out of the entire knowledge graph. The document and the remaining set of documents are then used as input for the document linker. This is repeated for all the documents in the training set. Because of the small dataset no cross validation was used. For each of these methods we hold on to the closed world assumption that if a link is not present within the given data set, it should not be a link.

### 6.1.1 Precision and recall

Precision and recall can be calculated when the complete system pipeline (including the threshold) is used. Precision reflects the fraction of relevant documents from all proposed documents and can thus be calculated as follows:

$$\text{precision} = \frac{|\text{ relevant documents} \cap \text{retrieved documents }|}{|\text{retrieved documents }|}$$

Within a recommender system, the precision indicates how sensible the proposed links seem to the user. A precision of 50% means that if for example two links are proposed to the user, at least one of them is correct. If one strives for a high user friendliness of the system, a high precision should be pursued.

Recall represents the fraction of relevant documents of all originally linked documents and can be calculated as follows:

$$\text{recall} = \frac{\mid \text{relevant documents} \cap \text{retrieved documents} \mid}{\mid \text{relevant documents} \mid}$$

The recall thus gives an indication of how well the sytem covers all the documents in the knowledge base. If the main goal of the system is to make the knowledge base as complete as possible, without taking user friendliness into account, a high recall should be obtained.

The F1-measure can be used to capture both precision and recall into one number. If precision and recall are equally important, this can be calculated in the following way:

$$\text{F1-measure} = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}}$$

All these metrics can be unraveled into precision, recall and F1-measure per document type to give more insight into the performance of the algorithms with regards to different document types. All of these metrics will be reported in terms of averages over the entire knowledge graph, as calculated using the earlier discussed ‘take one out principle’.

### 6.1.2 K-links

The k-links metric is used to evaluate the algorithms without being influenced by the threshold for the number of proposed links. For a document with a given number of correct links, it proposes the same amount of links that the document is known to have. This evaluation metric thus makes the assumption that the algorithm knows in advance how many links should be returned. By doing so, the recall and precision are equivalent since the number of relevant and retrieved document is the same. It prevents the precision from being too optimistic, which would be the case if the fixed number would be lower than the actual amount of links. It also prevents the recall for being too optimistic in the cases that the actual amount of links is lower than the fixed number of proposed links.

The disadvantage of the k-links metric is of course that it does not take into account the certainty the algorithm has due to the distances. For example, it could be that the distance of the first two ranked documents is very small, but the distance of the third is very large. If the original document has 10 links, the system is forced to additionally return the nine documents, even though these are likely to be wrong because they have a relatively big distance.

## 6.2 Distance evaluation

The results of the k-link measurements for all documents with each of the distance metrics is shown in table 2. This table shows that of all distance metrics, cosine seems to perform best on all vectorizers except for the glossaries of tags. The correlation metric is a close follow-up with approximately the same accuracies, which makes sense since these two metrics are very similar. The intersection vectorizer does not seem to be of much use, since it returns the same documents for each vectorizer. The eucledian distance has an accuracy of only 3.25% and 1.08% on the text and weighted text vectorizers respectively. It performs a bit better on the simple tag vectorizers and glossaries of tags (16.56% and 14.75% respectively) but still performs lower than the cosine metric. The bhattacharyaa vectorizer performs best on the glossaries of tags ans weighted tag vectorizers, though the difference with the cosine metric is not as high. Due to implementation difficulties, the tag smoothing vectorizer could not be measured with the eucledian distance and the simple tag vectorizer did not work for the bhattacharyya vectorizers.

## 6.3 Text-based descriptors

Table 2 shows the k-link values of all vectorizers, including those of the textvectorizer and the weighted-text vectorizer. The best performance of the textvectorizers is obtained by the cosine distance. This can be attributed to the fact that the cosine distance is independent to document length and only computes a similarity in document structure. In other words, when a document has the exact same words as a seconds document but only twice as much the cosine similarity classifies these documents as exactly equal. As the textvectorizer encodes the number of word occurrences in a vector the cosine distance can easily find document that use the same words frequently.

The cosine distance metric gives the best results for the text vectorizers. On average, 19.49% and 19.59% percent of the number of proposed links respectively are correct. The weighted text vectorizer performs a bit better, which is mainly due to an improvement in performance on information and questions.

Both textvectorizers have a low percentage correct with regards to proposing links for Persons. A further analysis shows that 76.36% (not weighted) and 69.09% (weighted) of the links for Persons are towards other Persons. However, within the Starfish network such links almost never occur (see table 6 for the distribution of document types within Starfish). This could explain the low performance of the textvectorizer on persons.

Overall, both textvectorizers are slow in performance even though the corpus is small. Additionally, the the bag-of-words approach imposes a few limitations on the document linker. Firstly, it performs bad when different languages are used. In the case of two different languages, there are less words that the two documents have in common. If important keywords entail word such as 'clickers' versus 'stemsysteem', there is no way of relating the two documents. Secondly, the current StarFish network consists of mainly textual content. However, in the future this is likely to be extended with images, videos and other non-textual content. These sources should then somehow be converted to text.

## 6.4 Tag-based descriptors

### 6.4.1 Simple tag similarity vectorizer

The performance of the simple tag similarity vectorizer, as shown in table 2 together with the other tag vectorizers, is about 26% precision when measuring k-link. The unraveling per document type shows that Question documents and Person documents perform the worst. This can be explained by the fact that half of both Questions and Persons have zero tags. Obviously, the simple tag vectorizer cannot deal with such documents. In fact, almost all other Questions have only one tag. Since the simple tag vectorizer compares vectors, it will prefer documents that also have only that particular tag, which makes it sensitive to attaching Questions to Questions. Something similar seems to happen with Persons, of which 50.91% of the connections are with other Persons. Apparently, persons with similar expertise are tagged similarly. However, as mentioned with the text vectorizer, in Starfish persons almost never refer to other persons. Moreover, if a document is badly labeled this can also induce problems. For example, the question 'TurnitIn licence' at the 'UvA' has two tags, but the simple tag similarity is unable to find both it's links because the link 'What is TurnitIn?' is not tagged with 'TurnitIn'. The text vectorizer, on the other hand, returns both links correctly. Good practices, events and projects perform better, but these document

| CORRELATION | Inf. | Question | Good Pr. | Project | Person | Event | Average |
|---|---|---|---|---|---|---|---|
| Textvectorizer | 20.49 | 42.02 | 30.35 | 25.41 | 5.81 | 21.42 | **19.73** |
| Weighted text | 21.25 | 44.82 | 30.35 | 16.04 | 5.81 | 21.43 | **19.76** |
| Simple tag | 21.33 | 16.67 | 69.64 | 37.81 | 14.87 | 46.83 | **21.90** |
| Tag smoothing | 20.95 | 21.93 | 44.64 | 31.04 | 13.59 | 46.83 | **20.78** |
| Glossaries of tags | 17.70 | 16.67 | 48.21 | 31.46 | 7.95 | 40.47 | **16.77** |
| Weighted tag | 17.70 | 16.67 | 48.21 | 31.46 | 7.95 | 40.48 | **16.77** |
| | | | | | | | |
| **COSINE** | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
| Textvectorizer | 20.49 | 40.70 | 30.36 | 25.42 | 5.81 | 21.43 | **19.49** |
| Weighted text | 21.67 | 44.82 | 30.36 | 16.04 | 5.81 | 21.43 | **19.90** |
| Simple tag | 21.21 | 16.67 | 69.64 | 37.81 | 17.53 | 46.83 | **22.80** |
| Tag smoothing | 20.58 | 21.93 | 44.64 | 31.04 | 13.59 | 46.83 | **20.69** |
| Glossaries of tags | 18.68 | 16.67 | 48.21 | 31.46 | 10.51 | 40.48 | **18.02** |
| Weighted tags | 18.68 | 16.67 | 48.21 | 31.46 | 10.51 | 40.48 | **18.02** |
| | | | | | | | |
| **INTERSECTION** | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
| Textvectorizer | 4.16 | 3.51 | 10.71 | 19.06 | 0.00 | 24.21 | **4.45** |
| Weighted text | 4.15 | 3.51 | 10.71 | 19.06 | 0.00 | 24.21 | **4.45** |
| Simple tag | 4.15 | 3.51 | 10.71 | 19.06 | 0.000 | 24.21 | **4.45** |
| Tag smoothing | 4.15 | 3.51 | 10.71 | 19.06 | 0.00 | 24.21 | **4.45** |
| Glossaries of tags | 4.16 | 3.51 | 10.71 | 19.06 | 0.00 | 24.21 | **4.45** |
| Weighted tag | 4.15 | 3.51 | 10.71 | 19.06 | 0.00 | 24.21 | **4.45** |
| | | | | | | | |
| **EUCLIDEAN** | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
| Textvectorizer | 3.04 | 3.95 | 17.86 | 0 | 0.85 | 21.43 | **3.25** |
| Tag smoothing | 1.32 | 1.32 | 7.14 | 0 | 0.43 | 0 | **1.08** |
| Simple tag | 14.05 | 10.53 | 30.36 | 38.23 | 14.36 | 35.71 | **16.56** |
| Tag smoothing | - | - | - | - | - | - | **0** |
| Glossaries of tags | 15.64 | 10.53 | 41.07 | 31.46 | 7.95 | 40.47 | **14.75** |
| Weighted tags | 15.64 | 10.53 | 41.07 | 31.46 | 7.95 | 40.47 | **14.75** |
| | | | | | | | |
| **BHATTACHARYYA** | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
| Textvectorizer | 15.57 | 35.88 | 30.36 | 16.15 | 5.30 | 21.43 | **16.19** |
| Weighted text | 18.72 | 29.04 | 30.35 | 11.99 | 7.95 | 21.43 | **16.63** |
| Simple tag | - | - | - | - | - | - | **-** |
| Tag smoothing | 4.16 | 3.51 | 10.71 | 19.06 | 0 | 24.2 | **4.44** |
| Glossaries of tags | 18.36 | 17.98 | 48.21 | 48.13 | 10.51 | 40.45 | **19.39** |
| Weighted tag | 18.36 | 17.98 | 48.21 | 48.13 | 10.51 | 40.45 | **19.39** |

Table 2: This table shows the performance based on k-link measuring of all documents. This means that the numbers are the percentage of correctly returned links, given the constraint that the number of links that is returned is equal to the known number of links a document has. These are averages over the entire document base. The left upper corner of each table tells which distance metric was used. Each row describes the performance of one vectorizer. Each column describes the performance of each document type. Thus, the first number 20.49 means that the average percentage correctly returned links of documents of the type 'Information' for the textvectorizer using the correlation distance is 20.49%.

types only make up to 3.2%, 2.7% and 5.4% of the total amount of documents respectively so have less influence on the average precision.

#### 6.4.2 Tag smoothing

The performance of the simple tag vectorizer, as shown in table 2, is quite similar with the results of the tag similarity. It performs worse on the information. This is visible in the 'Peer-instruction' information document, in which the tag smoothing fails to make a recommendation the simple tags do. It also performs worse on Persons, which is visible in the 'Claire McDonnell' person document. That document has more tags than average but does not return any correct links.

In the current implementation this vectorizer is relatively slow. In practice the similarity matrix can be pre calculated and updated in batches. Due to the transform on the tag similartity matrix, it is very hard to determine which tag occurrences contributed to the document similarity and why some recommendations are made. It does not seem to perform much better than the regular bag of words tag descriptor, in Zhou et al. the algorithm only starts performing significantly better when it is presented with more tags.

#### 6.4.3 Glossaries of tags

The glossaries of tags approach returns the lowest results without applying the threshold. This could be a result of the sheer number of tags which have a glossary. For example, the documents 'Is there an English version in Tentamenlade?', 'De toetscyclus' and 'Wat is het verschil tussen Learning Analytics en TTL?' all have the tag 'ToetsenEnToetsgestuurdLeren' and at most one other tag with a glossary. ToetsenEnToetsgestuurdLeren is a tag with a glossary, but that glossary is one sentence long. This results in weak vectors which aren't very well distinctive from other documents which have little tags or little tags with glossaries. The information needed to link the documents does not seem to be present in the current dataset.

#### 6.4.4 Weighted tag glossaries

Table 2 shows that the weighted tag vectorizer, an extension of the glossaries of tags vectorizer, performs exactly the same as the glossaries of tags vectorizer. It thus seems that applying the weights has no influence on the performance of this vectorizer. The histogram in figure 4 shows the number of tags that appear in a certain amount of documents. From this can be derived that there are many tags that almost all the tags appear in about 13 documents or less. There are only a few tags that appear in a lot of documents. Thus, the small influence of the weighted tag vectorizer is caused by the fact that many tags assign the same or a similar weight to vector. This could explain why it proposes the same links as the glossaries of tags vectorizer.

#### 6.4.5 Hybrid

Given the previous results, it can be concluded that the text vectorizer works better on Questions than the tag vectorizers, though it performs less good on other document types. This difference is probably due to the fact that many Questions have no tags. The hybrid vectorizer takes the best of both worlds by combining the textvectorizer and simple tag similarity into one vectorizer. The textvectorizer was used instead of the weighted text vectorizer because they differed only little in performance, but the textvectorizer is a bit faster. If a document has no tags, it will be handeled by the textvectoriser, otherwise the simple tag similarity will propose links. This vectorizer works best if cosine distance is used. These results are shown in table 3. This table shows that the hybrid vectorizer performs significantly better with an accuracy of 26.13% than the text and tag

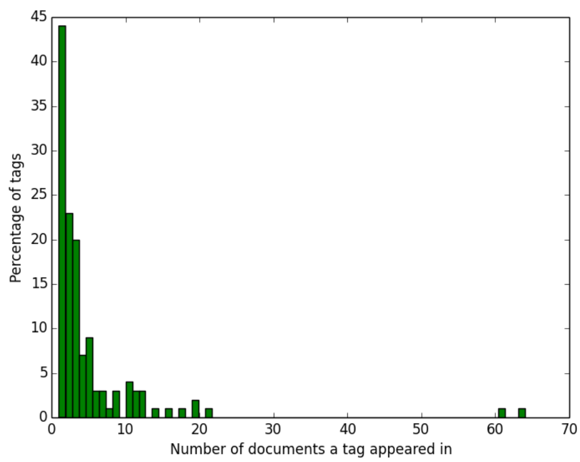


Figure 4: This histogram shows the number of tags that appear in a certain amount of documents. Almost all tags appear in 10 documents or less. Only a few tags are very common and these appear in about 60 different documents.

vectorizers themselves, which had an accuracy of 19.49% and 22.80% respectively. The percentage of correctly answered questions, 31.93%, is much higher than that of the simple tag vectorizers. For the other document types, the accuracy is comparable with that of the simple text vectorizer. Thus, the hybrid vectorizer indeed improves accuracy on Questions.

| | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
|---|---|---|---|---|---|---|---|
| Accuracy hybrid | 21.21 | 31.93 | 69.64 | 37.81 | 19.15 | 46.83 | **26.13** |

Table 3: This table shows the accuracy of a vectorizer that is a hybrid between the textvectorizer and the simple tag vectorizer using cosine distance. It shows the average percentage of correctly returned links for a given document type if the known number of correct links is proposed by the vectorizer.

## 6.5 Bayesian weighting

Table 4 shows the performance of each of the vectorizers (all with cosine distance) while applying the tag and link devaluation using the k-link metric. It shows that the performance of all algorithms drastically decreases when the probabilities are used to re-rank the documents. Table 4 shows the distribution of links in the simple tag similarity vectorizer, with and without the probabilities (all based on k-link measuring). It is clear that the links with probabilities have a sharper distribution: the sparseness of the table shows that many types of links do not even exist. There seem to be some correlations between the vectorizer with probabilities and the real distribution of links. The original distribution of links in the current Starfish knowledge base is shown in table 5. The preferred effect of having no Persons link to other Persons, which was mentioned as a flaw of both the tag and text vectorizers, was done correctly. However, the Information documents are all linked to other Information documents. Table 5 shows that Information documents indeed often are linked to other Information documents, but they also

| DEVALUATION | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
|---|---|---|---|---|---|---|---|
| Textvectorizer | 0.59 | 12.63 | 0.00 | 0.00 | 0.43 | 0.00 | **2.58** |
| Weighted text vectorizer | 0.59 | 16.58 | 0.00 | 0.00 | 0.43 | 0.00 | **3.29** |
| Simple tag similarity | 1.18 | 10.53 | 0.00 | 4.17 | 0 | 0 | **2.55** |
| Tag smoothing | 0.00 | 10.53 | 0.00 | 4.17 | 0.43 | 0.00 | **2.33** |
| Glossaries of tags | 4.61 | 13.16 | 0.00 | 6.23 | 6.41 | 0 | **6.61** |
| Weighted tags | 4.61 | 13.16 | 0.00 | 6.23 | 6.41 | 0 | **6.61** |

Table 4: The average k-link accuracy per vectorizer per document type if both tag and link devaluation are used to re rank the documents. All vectorizers used the cosine distance.

| DEVALUATION | **Inf.** | **Question** | **Good Pr.** | **Project** | **Person** | **Event** |
|---|---|---|---|---|---|---|
| **Information** | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Question** | 11.11 | 84.44 | 0 | 2.22 | 2.22 | 0.00 |
| **Good Practice** | 58.82 | 0.00 | 41.18 | 0.00 | 0.00 | 0.00 |
| **Project** | 93.33 | 3.33 | 0.00 | 0.00 | 3.33 | 0.00 |
| **Person** | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **Event** | 21.43 | 0.00 | 0.00 | 21.43 | 0.00 | 57.14 |
| | | | | | | |
| NO DEVALUATION | **Inf.** | **Question** | **Good Pr.** | **Project** | **Person** | **Event** |
| **Information** | 21.21 | 16.67 | 69.64 | 37.91 | 17.44 | 46.83 |
| **Question** | 6.67 | 35.56 | 17.79 | 4.44 | 35.56 | 0 |
| **Good Practice** | 41.18 | 11.76 | 17.65 | 5.88 | 5.88 | 17.65 |
| **Project** | 30.0 | 16.67 | 6.67 | 13.33 | 20.00 | 13.33 |
| **Person** | 30.01 | 7.27 | 3.64 | 1.81 | 50.01 | 5.45 |
| **Event** | 35.71 | 0.00 | 14.29 | 28.57 | 7.14 | 14.29 |

Table 5: This table shows the percentage of links from one type (row) to another (column) for simple_tag_vectorizer with tag and link devaluation (above) and without (below), measured using k-link. The rows sum up to 100%. Thus, the first value 100.0% means that if tag and link devaluation is performed, all proposed links for all newly added documents of the type Information are themselves also Information documents.

often link to other Persons, which is not captured by the vectorizer that uses probabilities. If the vectorizer does not use the probabilities, the distribution is less sharp, as shown in the lower half of table 4. This distribution seems to have no correlation with the real document distribution.

Figure 5 gives insight into the reason why the probabilities do not improve the performance, even though they offer a distribution of links between document types that is closer to the real distribution. The figure shows a histogram of the percentage of documents that have a certain probability. The left side shows the distribution for the tag probabilities, where the red bars represent incorrect links and the green bars show the correct ones. These numbers are obtained by making the simple tag vectorizer propose as many links as there are documents. One would expect that a higher percentage of correct links would be on the righthand side of the histogram, since these should have a higher probability. However, this is clearly not the case. On the contrary, about 75% of the incorrect links have a chance of 0.0014, the highest probability. The link probability, shown on the right hand side of the figure, is a bit more promising since the incorrect links are a bit higher on the left hand side of the histogram. However, there is still no clear difference in distribution between correct and incorrect links. This could also be an explanation of the bad performance of the vectorizers that use probabilities, besides the sharp

| | Inf. | Question | Good Pr. | Project | Person | Event |
|---|---|---|---|---|---|---|
| **Information** | 39.86 | 8.39 | 4.20 | 5.59 | 37.76 | 4.20 |
| **Question** | 19.72 | 25.35 | 12.68 | 12.68 | 23.94 | 5.63 |
| **Good Practice** | 25.00 | 17.86 | 7.14 | 10.71 | 25.00 | 14.29 |
| **Project** | 23.91 | 10.87 | 4.35 | 19.57 | 30.43 | 10.87 |
| **Person** | 69.64 | 8.93 | 1.79 | 8.93 | 1.79 | 8.93 |
| **Event** | 28.57 | 0.00 | 14.29 | 9.52 | 33.33 | 14.29 |

Table 6: This table shows the percentage of links from one type (row) to another (column) for the links as they are in the document base of Starfish. Thus, the first value 39.86% means that 39.86% of the proposed links for Information documents are of the type Question.

distribution that was discussed before.

## 6.6 Threshold performance

To evaluate the threshold, it is important to know what the result of the nearest neighbor algorithm is and how this relates to the way experts decide a document is relevant or not. Figure 6 shows the increase of distance for neighbors when ranked less similar then their preceding neighbor.

Documents with little links registered in the original dataset are blueish in figure 6 and expected to have a small set of neighbors close by, after which the distance for the following neighbors should rise quickly. This general trend is visible in almost all tag vectorizers.

Documents with more than average links registered in the original dataset are purplish in figure 6 and the neighbor distance is expected to increase slower than the blue lines.

The results shown are very noisy, something that is to be expected from a small human annotated dataset. However, we can detect anticipated general patterns. Figure 7 shows the differences between consecutive neighbors. In line with our understanding of expert linking, these graphs take the form of a long tail distribution for documents with a small amount of links.

The above evidence suggests the nearest neighbor algorithm returns neighbours at distances in a similar to how experts would rate similarity. The reported decrease of distance differences is expected and should be handled properly by the proposed threshold calculation.

Knowing that the pipeline until thresholding returns sane results, the actual performance of the proposed threshold formula is tested on the dataset. The performance of the automated threshold was measured for all vectorizers with the cosine metric since that one gave the overall best reuslts. Now - in contrast with the k-link measurements, precision, recall and the F1-measure can be calculated because the threshold can return a different number of links than the links that are known to be correct. Table 7 shows the precision, recall and f1-measure for each of the vectorizers, including an unraveling for each type of document. A comparison between this table and table 2 and table 3, which were measured with the k-link metric, shows that the f1-measure of most outcomes is comparable with the k-link accuracy that was measured. Since these two metrics are calculated differently, we cannot compare the results directly. However, the differences between the different vectorizers are approximately the same. For the tag vectorizers, precision and recall are in approximately the same range. This indicates that the threshold does not have a clear preference for eighter returning too many (which would result in a higher recall) or too little (which would return in a higher precision) links. However, the textvectorizers seem to have a preference for returning too little documents, with a recall and precision of respectively 15.66% and 15.11%. The glossaries of tags approaches seem to have a preference for a higher recall, with 38.80% and 17.26% for recall and precision respectively. This seems to be caused by a much higher precision and recall for the Person type of documents. In the knowledge base, a Person

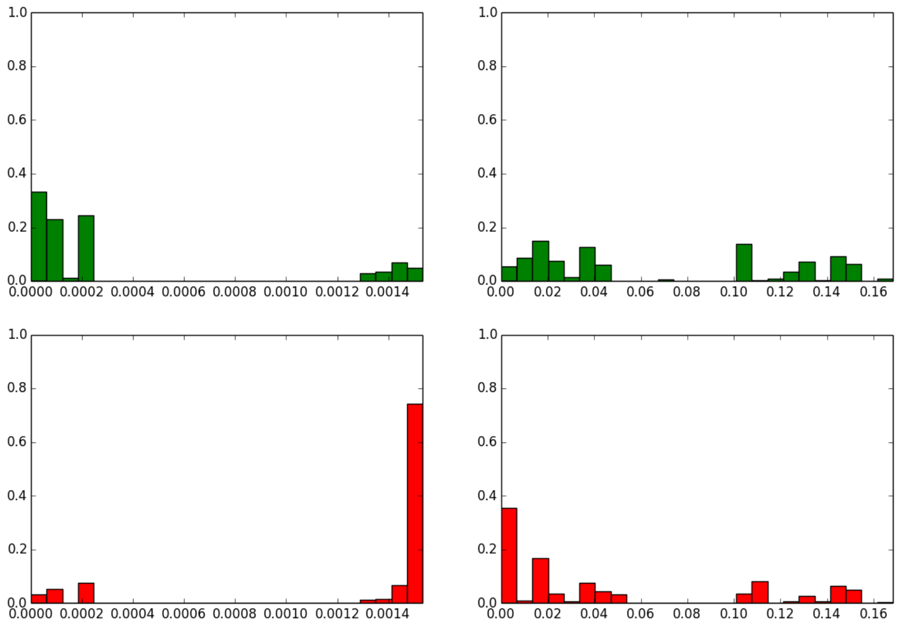


Figure 5: The distribution in percentages of correct (green) and incorrect (red) document links as proposed by a simple tag vectorizer that proposed all the documents in the dataset. Left hand side shows the tag probabilities and the right hand side the link probabilities. Thus, about 30% of the correct links as proposed by the simple tag vectorizer have a probability between 0 and 0.00005.

| THRESHOLD RECALL | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
|---|---|---|---|---|---|---|---|
| Textvectorizer | 12.05 | 39.65 | 19.64 | 20.73 | 5.21 | 21.43 | **15.66** |
| Weighted text | 14.80 | 29.82 | 19.64 | 24.37 | 5.24 | 21.43 | **15.11** |
| Simple tag | 24.14 | 20.61 | 48.21 | 30.63 | 19.49 | 26.19 | **23.26** |
| Tag smoothing | 26.20 | 21.93 | 55.35 | 23.85 | 22.05 | 46.83 | **25.42** |
| Glossaries of tags | 36.51 | 23.25 | 21.43 | 36.35 | 50.20 | 44.05 | **38.80** |
| Weighted tag | 36.52 | 23.25 | 21.43 | 36.35 | 50.20 | 44.05 | **38.80** |
| Hybrid | 24.14 | 41.40 | 48.21 | 30.63 | 20.34 | 26.19 | **27.26** |
| THRESH PRECISION | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
| Textvectorizer | 26.47 | 50.00 | 41.67 | 24.79 | 24.79 | 7.26 | **24.05** |
| Weighted text | 26.76 | 41.67 | 33.33 | 26.25 | 8.55 | 27.78 | **23.00** |
| Simple Tag | 28.43 | 17.54 | 62.60 | 56.25 | 10.77 | 38.89 | **23.71** |
| Tag smoothing | 23.77 | 17.11 | 50.00 | 43.75 | 8.38 | 50.00 | **20.19** |
| Glossaries of tags | 25.00 | 14.47 | 37.50 | 33.18 | 17.26 | 46.67 | **22.00** |
| Weighted tag | 25.00 | 14.47 | 37.50 | 33.18 | 17.26 | 46.67 | **22.00** |
| Hybrid | 28.43 | 39.91 | 62.50 | 56.25 | 13.33 | 38.89 | **28.61** |
| | | | | | | | |
| THRESH F1-MEASURE | Inf. | Question | Good Pr. | Project | Person | Event | **Average** |
| Textvectorizer | 14.92 | 37.59 | 26.66 | 20.03 | 5.60 | 22.72 | **16.59** |
| Weighted text | 16.53 | 31.47 | 23.06 | 23.24 | 6.58 | 23.33 | **16.57** |
| Simple tag | 23.53 | 17.31 | 44.72 | 34.61 | 12.56 | 31.17 | **20.27** |
| Tag smoothing | 22.04 | 17.54 | 45.23 | 29.76 | 11.26 | 48.29 | **19.49** |
| Glossaries of tags | 16.74 | 20.75 | 38.26 | 28.28 | 9.49 | 44.44 | **17.26** |
| Weighted tags | 16.74 | 20.75 | 38.26 | 28.28 | 9.49 | 44.44 | **17.26** |
| Hybrid | 23.53 | 35.31 | 44.72 | 34.61 | 13.85 | 31.17 | **23.92** |

Table 7: This table shows the precision, recall and F1 measure for all vectorizers per document type if they are thresholded with the parameter 0.3. All vectorizers used cosine distance. Thus, the first value 12.05 menas that the textvectorizer had a 12.5% average precision documents of the type Information.

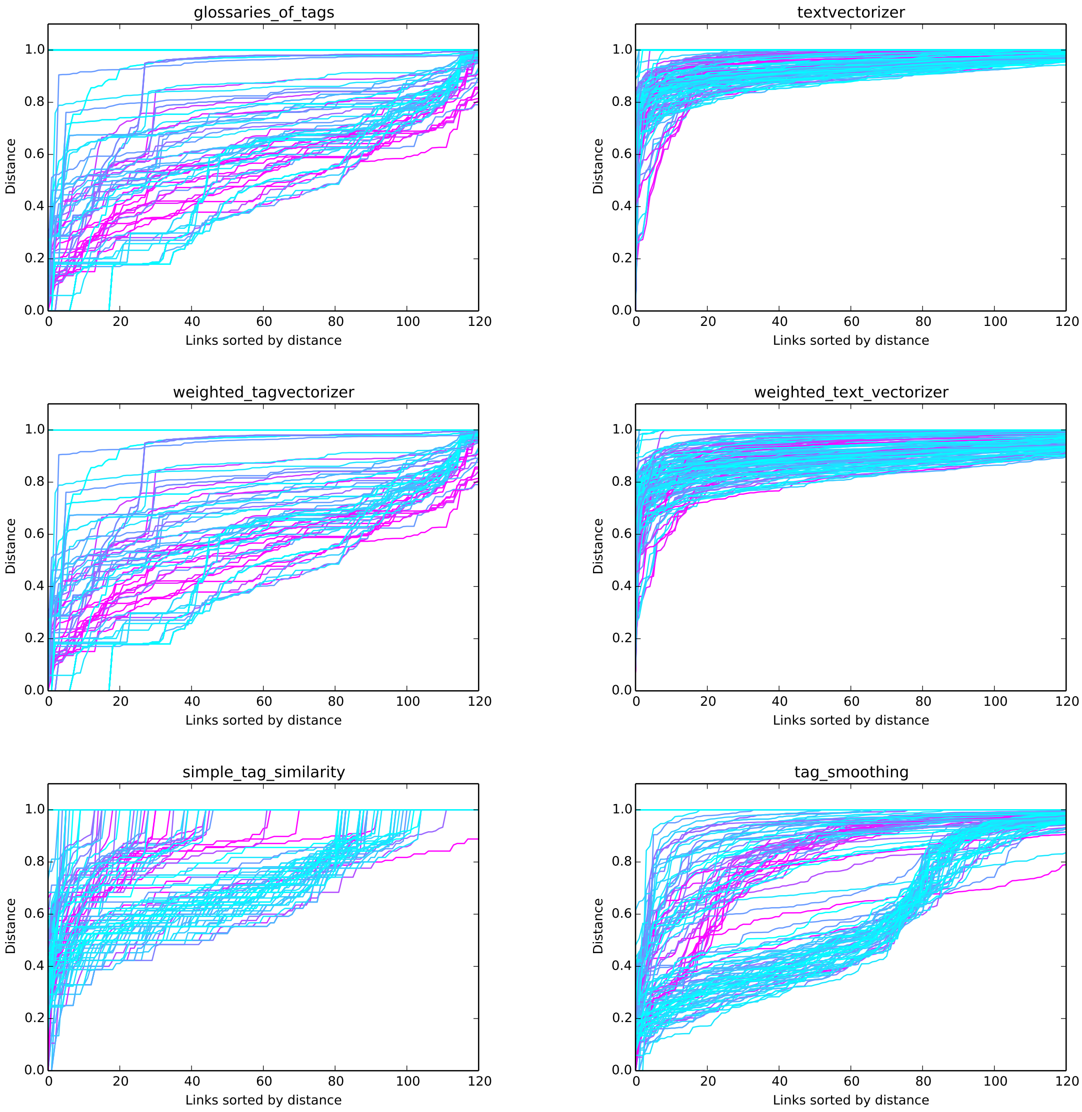


Figure 6: The sorted cosine distances of nearest neighbors and their distance for differend vectorizers. The blue purple gradient represents documents with 0 links (blue) to 10 links (purple)

has on average about 1.6 links. However, the thresholded glossaries of tags returns 5.9 links on average. For example, the Person Andre Heck is linked to two documents, an Event and a Project. The k-link vectorizer returns two false documents - a Good Practice and a Qeustion. The thresholded vectorizer returns a total of 5 links, including the correct Event. It depends on a preference for recall or precision whether or not this difference in performance is desirable.

Of all vectorizers, the hybrid vectorizer still performs the best. Precision and recall are 27.26% and 28.61% respectively. The reason why the F1-measure is lower than these two, is due to the fact that we are dealing with an average measurement. Apparently, there is quite often a

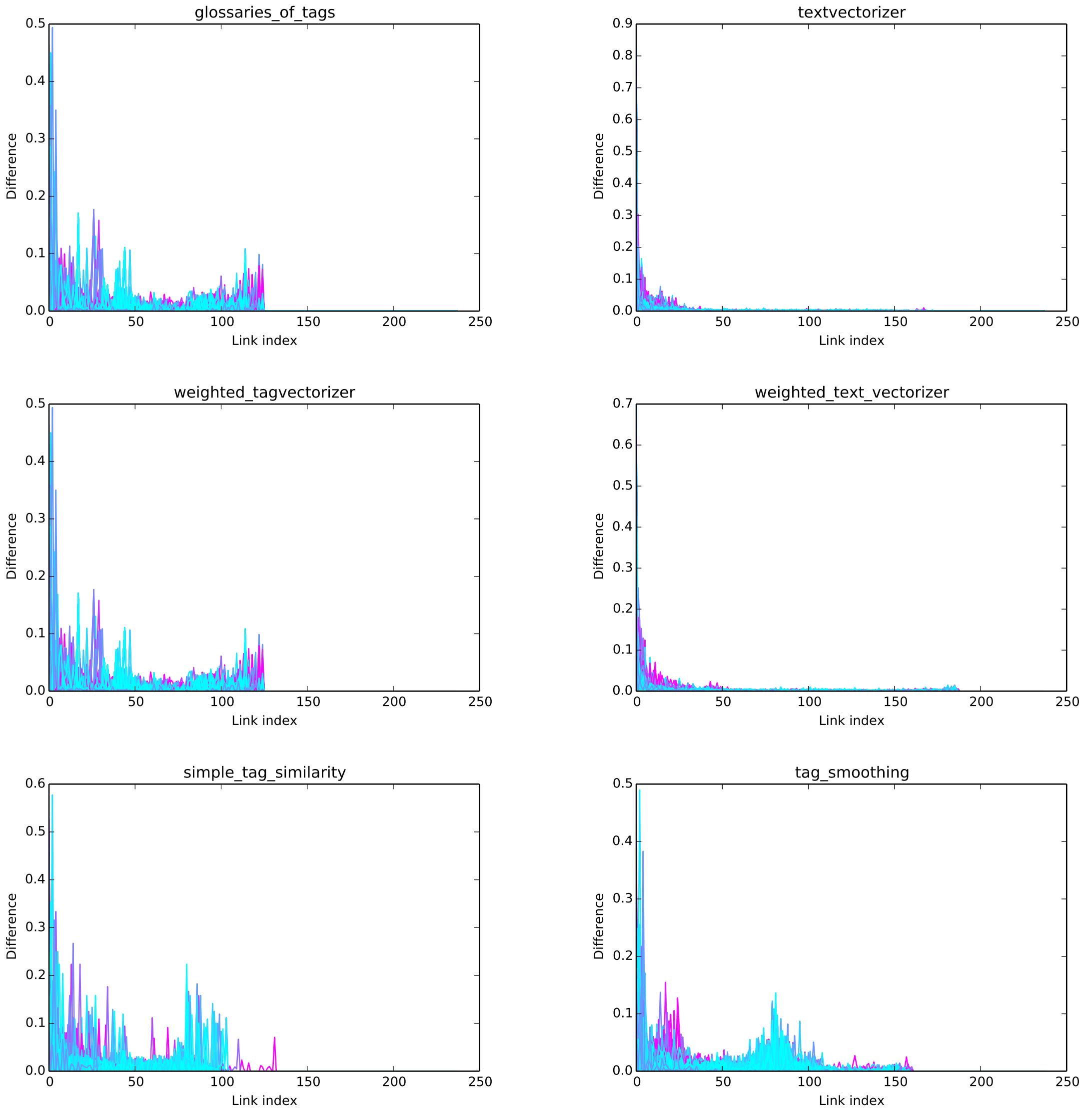


Figure 7: Cosine distance differences betweeen two nearest neighbors sorted by document rank. The blue purple gradient represents documents with 0 links (blue) to 10 links (purple)

difference between precision and recall per document.

# 7 Conclusion

This report has compared seven different ways (vectorizers) of converting a text document into a vector representation and five ways of computing the distance between these vectors. All this is done in the context of the Starfish network. Also a method to take the knowledge from the network into account and a way to determine the number of documents to retrieve were proposed.

These combined form a complete 'pipeline' to compute and propose links to known documents for a new document that is about to be added to the Starfish network.

The following conclusions can be drawn from the present study: firstly, the text vectorizers performs the best if the newly added document is a Question (42.02-44.82% accuracy on k-link measuring). However, it cannot deal with documents that have different languages or non-textual documents such as images, videos and audio. The simple tag vectorizer has the best performance (22.80% overall average accuracy on k-link measuring) and is the fastest. The best overall performance with the k-link measurement is gained with the hybrid vectorizer (26.13%) that uses the textvectorizer if no tags are available and the simple tag vectorizer otherwise. This vectorizer performs as good on most document types as the simple tag vectorizer, but performs significantly better on questions (31.93% versus 16.67%). Secondly the probabilistic model of the network that is proposed is either to simplistic or the data available is too little. In either case it might be off interest to further investigate a similar model on a bigger data set. Lastly the method of selecting the number of documents shows that the overall performance does not change significantly if the threshold is added to most vectorizers. However, the text vectorizers seem to have a bias towards a higher precision in the trade off between precision and recall. The glossaries of tags and weighted glossaries of tags get a higher recall for persons. The best performance while using the threshold was obtained using the hybrid vectorizer, with an average precision, recall and F1 measure of respectively 28.61%, 27.26% and 23.92%.

The findings in this report are subject to at least three limitations. First, the proposed solution only works for textual document and not on audio, video which may be part of the Startfish knowledge graph. Secondly, the data set that was used during investigation is rather small and may not be representation of a real life data set. Lastly due to the small training network no cross-validation was performed which may result in an overfit to the data.

Whilst this study is based on only a small network of documents taken together these findings do show that the chosen solution is capable of recommending links between documents with a certainty far above the guessing level. It is now up to the client to choose if a precision of 28.61% is good enough to let the user select a document to link and if a recall of 27.26% covers enough of the documents within the knowledge base.

# 8 Future Work & Recommendations

Current work focusses in creating outgoing links from a new document when it is inserted into the network. However, due to the directed structure of the network graph, this means a new document will only receive links of documents inserted after itself. The Starfish system could benefit even more from link recommendation if it also proposes incoming links. This will ensure a graph in which all related documents are even better connected then they are now. Most methods proposed in this research are based on symmetrical similarities, future work investigating Starfish specific properties to distinguish between incoming and outgoing links would therefore be very interesting.

The current system is implemented on a graph with relatively small size, but this research provides a foundation to further investigate linking in Starfish. It is recommended to verify the results of this project as Starfish grows, to ensure the current solution is not prone to overfitting.

Meanwhile, the sparsity of tags for the tag based recommendations could be addressed by applying topic modeling. Automatically generated topics from algorithms such as Latent Dirichelet Allocation or Explicit Semantic Analysis could regular tags in the current system.

# References


Chaudhuri, G. (2011). Bhattacharyya distance. *Encyclopedia of Mathematics (www.encyclopediaofmath.org)*, (1).

Park, J., Choi, B.-C., and Kim, K. (2010). A vector space approach to tag cloud similarity ranking. *Information Processing Letters*, 110(12):489–496.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International journal of computer vision*, 7(1):11–32.

Zhou, J., Nie, X., Qin, L., and Zhu, J. (2011). Web clustering based on tag set similarity. *Journal of computers*, 6(1):59–66.