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 Supervisor	Robrecht Jurriaans
	Sander Latour
	Wijnand Baretta

June 25, 2014
Universiteit van Amsterdam

Abstract

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 the products of Perceptum, 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, a question. Each document has a set of tags associated with it, which describe the different aspects of educational innovation. 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, which can devote more time in getting to know all the documents. However, that approach has two main drawbacks. First of all, this would mean that some central authority determines whether or not two documents should be linked. This is not in line with the idea of a community-driven knowledge base. Secondly, if the knowledge base grows even further, 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 perfect, but should work reasonably 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	Starfish users
who	search for and edit knowledge in starfish
the	starfish document linker
is	a starfish core system addition
that	finds related documents
unlike	moderated or individual linking
our	product uses algorithms and data to 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 becomes a problem. Starfish tries to overcome this by structuring it's knowledge base in a non-hierarchical manner. There are subcommunities 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 variance 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, title, text, tags and links. The Person type is an exception on that and has a name-field instead of title and a about-field instead of text. 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 x outgoing links, and x incoming links.

Each document in the knowledge base has a set of tags assigned to it, based on the different aspects of educational innovation the document covers. On average a document has x tags. Glossaries are special types of documents, as they are description for tags. This means that it is unneccessary for the link recommendation system to return Glossaries, since a glossary could better be 'linked' via assigning a 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, x unique tag concepts can be distinguished of which x% 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 peformed 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. Though Starfish aims to be user-driven, it is unfeasible wiht the current dataset to make the linking user driven. Such an approach could focus on a reinforcement learning system, that learns of the the opinion of users by means of a voting procedure. However, the data needed for this is simply not available.

3 Theory

In solving the linking problem, one must find a way to compare different documents based on their linkability with a newly added documents. Computationally, this can be done by creating document descriptors - vectors that the describe the features of a document in some way. Given the Starfish domain, this can be done in a tag-based and text-based approach towards creating document descriptors. This section wil 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:

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d) : w \in d\}}$$
$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$
$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

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 inversed 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 Nearest Neighbor and distance metrics

The

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. Firstly, each of the documents is transformed into a descriptor. Secondly, 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 Nearest Neighbor algorithm with several distance metrics. Lastly, an algorithm chooses the proper amount of proposed links that must be returned.

```
python documentlinker.py −vectorizer <vectorizername>
−distance <distance metric>
−bayes <true/false>
−threshold <0..1>
```

We will now discuss each of these parameters, since these will give more insight into the approach that was chosen to solve the problem. For the performance of the different parameters we refer to the experiment section.

4.1 Vectorizer

The first step is to create document descriptors, which is done by algorithms that we call *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 increased with the sum of the descriptors of the document it is linked to itself, decreased by some weight. 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 they make use of the tags that are assigned to the documents. 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 exists of documents with associated tags like $\{\{t_1, t_2\}, \{t_1, t_3\}\}$. From the co-occurence it does not follow that t_2 and t_3 are related, however by transitivity with t_1 we want to create a small implicit link between t_2 and t_3. The tag smoothing method does this based on work from **?**.

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 indicate the similarity 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.

4.2 Distance

A ranking is created using the Nearest Neighbor algorithm that sorts the document descriptors based on their distances with the newly added document. The following five distance metrics were implemented:

- 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.

4.3 StarFish specific adaptations: Bayesian weighting

Both the tag-based and text-based approaches uses 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 is almost never linked to another person. In the Bayesian weighted vectorizer this is captured by weighting the vectors with 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 this probability is multiplied with the distances that come from Nearest Neigbor 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 the 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 for Starfish.

4.5 Results

The document linker saves the results for a given knowledge base to a file called $< vectorizer ><$ $distancemetric > .json$. The proposed links can be viewed by opening *view.html* from a browser and selecting the preferred json file.

5 Method

5.1 Data preprocessing

Bla.

5.2 Text vectorization

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, we define 'content' as the title and text-fields of a document. The only exception on this are Persons, of which we wil use the name and about-fields.

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 (**?**). 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 stopword 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 it's 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 algorithm is displayed in figure xx.

5.3 Tag vectorization

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 a document is really about. The simple tag vectorizer is a very straight forward 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 vectorizers creates descriptors based on the tag set of a document. A tag co-occurs with other tags in a document, we assume documents with similar tags should be linked in Starfish. Let the frequency of occurrence with other tags across the dataset will 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.

? 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 1. 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 v_i and v_j. These vectors cover both explicit similarity and implicit similarity (**?**). This transfers the problem to find a set of linearly independent vectors $\{v_1, v_2, \ldots, v_n\}$ for which for all $v_i \cdot v_j = \cos(\theta_{i,j})$. 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 **?** introduces a modified Cholesky transform.

$$s_{i,j} = \frac{f_{i,j}}{f_i + f_j - f_{i,j}} \tag{1}$$

5.3.3 Glossaries of tags

- Motivation: find underlying network between tags by using their glossaries
- Implementation: hybrid form of text and tag vectorizer

5.4 Distance metrics

Short description of how these were implemented

5.5 Bayesian weighting

Explain motivation, theory and implementation

5.6 Thresholds

Explan motivation, theory and implementation

6 Experiments

6.1 Evaluation metrics

In order to evalute the performance of the different algorithms, three different metrics were 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 is used. Precision reflects the fraction of relevant documents from all proposed documents and can thus be calculated as follows:

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

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

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

The precision and recall can be unraveled into precesion and recall per document type to give more insight into the performance of the algorithms with regards to different document types.

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 of 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.

6.1.3 Qualitative descriptor analysis

Based on the precision and recall (both with threshold as with k-links) a general analysis can be performed. To get more insight into the exact reasons why one particular algorithms performs in the way it does, a qualitative analysis was performed in which the proposed document descriptors and distances for several documents were compared with the descriptors of known correct links.

6.2 Text-based descriptors

Table x shows the k-link values of the text-based descriptions: the textvectorizer and the weighted-text vectorizer.

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. Figure x shows the differences of vectors of three texts when an English document is combined with an English proposed document and a Dutch proposed document. 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.

6.3 Tag-based descriptors

Due to it's simplicity, the simple tag vectorizer is very fast. It's performance, as shown in table xx, is about 24% precision. Both Question documents and Person documents perform quite bad. 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 wil 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 45% of the connections are with other Persons. Apparently, persons with similar expertices are tagged similarly. However, as mentioned with the tag vectorizer, in StarFish persons almost never refer to other persons. Moreover, if a document is badly labeled this can also induce problems. For example, take the question 'Is there an English version in Tentamenlade', tagged with 'ToetsenEnToetsgestuurdleren'. The proposed links are visible in table xx, which shows that the three proposed questions all have the tag 'ToetsEnToetsGestuudLeren'. However, if the question was tagged with the tag 'Tentamenlade', which seems very reasonable given the proposed question, the false negatives would likely be returned correctly by the system. Good practices, events and projects perform significantly better with 75,6%, 73,0% and 35,8%. These are often thoroughly tagged, as shown by document xx, which has a rich set of tags. However, these document types only entail 3.2%, 2.7% and 5.4% of the total amount of documents respectively, which explains why the total performance is still stuck at 24.8%.

```
False Positives:
- Wat is het verschil tussen Learning Analytics en TTL (Question)
- Formatieve meerkeuze toetsen om begin kennisniveau te toetsen (Good Practice)
- De toetscyclus (Information)

True Positives:
- Tentamenlade2.5 (Project)

False Negatives:
- Tentamenlade Natuurkunde (Information)
- Hoe kan ik inloggen in Tentamenlade? (Question)
- Hoe kan ik inloggen in Tentamenlade? (Question)
```

Evaluation in cijfers hier

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 **?** the algorithm only starts performing significantly better when it is presented with more tags.

6.4 Bayesean weighting

Table xx 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 1 shows the distribution of links in the vectorizer that performed best, the simple tag similarity vectorizer, with and without the probabilities (all based on k-link measuring). It is clear that the links with probabilties have a sharper distribution: the sparseness of the table shows that many

	Inf.	Question	Good Pr.	Project	Person	Event
Information	1.53	0.76	0.00	0.00	97.71	0.00
Question	1.72	27.59	0.00	1.72	68.97	0.00
Good Practice	0.00	0.00	0.00	0.00	100.00	0.00
Project	2.50	2.50	0.00	0.00	95.00	0.00
Person	100.00	0.00	0.00	0.00	0.00	0.00
Event	23.53	0.00	0.00	0.00	76.47	0.00

	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 1: 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%.

	Inf.	Question	Good Pr.	Project	Person	Event
Information	18.99	5.04	1.94	3.10	68.60	2.33
Question	14.00	23.00	6.00	7.00	50.00	0.00
Good Practice	11.11	8.89	4.44	4.44	64.44	6.67
Project	12.50	7.50	2.50	7.50	65.00	5.00
Person	85.19	3.70	0.93	4.63	0.93	4.63
Event	28.12	0.00	9.38	3.13	59.38	0.00

Table 2: Percentage of links from one type (row) to another (column) for the real links in the document base

types of links do not even exist. This effect could be caused by overfitting - the data set could be too small to calculate reliable probabilities. The distributions with and without probabilities can be compared with the original distribution of links in the current Starfish knowledge base, as shown in table 2.

Figure 1 gives insight into the reason why the probabilities do not improve the performance. 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. 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.

6.5 Threshold performance

The performance of the automated threshold was measured for all vectorizers using the cosine distance metric. Table xx shows the precision and recall for each of the vectorizers, including an

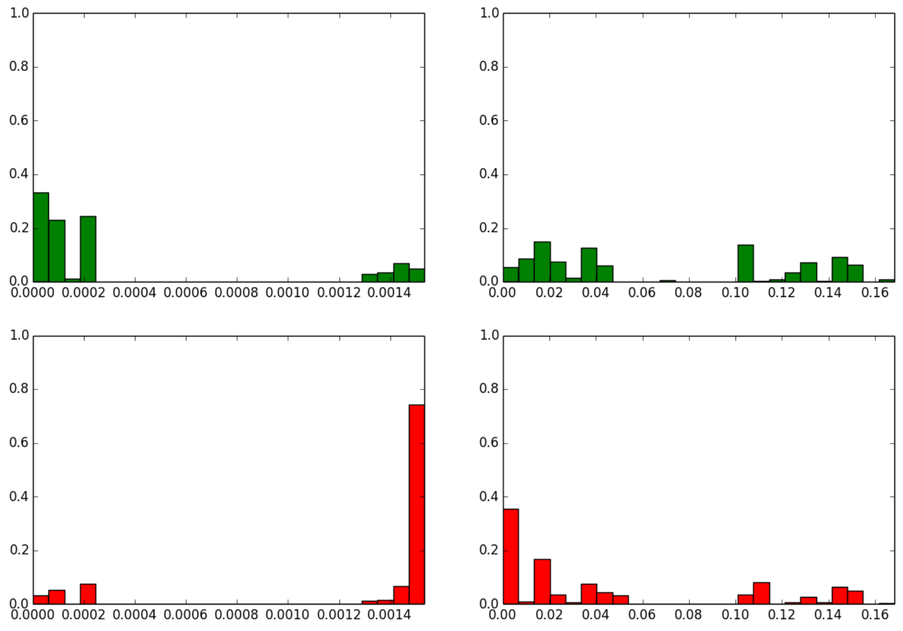

Figure 1: The distribution in percentages of correct (green) and incorrect (red) document links. Left hand side shows the tag probabilities and the right hand side the link probabilities.

unraveling for each type of document. A comparison between this table and table xx, which is measured with the k-link metric, shows that overall, performance does not decrease drastically when using the automated threshold. Setting the correct threshold is always a trade-off between precision and recall.

7 Conclusion

- Which vectorizers do not work
- Which vectorizers do work
- How well does the bayesian layer perform
- How well does the thresholds perform

8 Future Work & Recommendations

Latent dirichlet allocation because tags in dataset are not so good.

Also create incoming links