Effective Unsupervised Author Disambiguation with Relative Frequencies

Tobias Backes
GESIS - Leibniz-Institute for the Social Sciences
tobias.backes@gesis.org

ABSTRACT

This work addresses the problem of author name homonymy in the Web of Science. Aiming for an efficient, simple and straightforward solution, we introduce a novel probabilistic similarity measure for author name disambiguation based on feature overlap. Using the researcher-ID available for a subset of the Web of Science, we evaluate the application of this measure in the context of agglomeratively clustering author mentions. We focus on a concise evaluation that shows clearly for which problem setups and at which time during the clustering process our approach works best. In contrast to most other works in this field, we are skeptical towards the performance of author name disambiguation methods in general and compare our approach to the trivial single-cluster baseline. Our results are presented separately for each correct clustering size as we can explain that, when treating all cases together, the trivial baseline and more sophisticated approaches are hardly distinguishable in terms of evaluation results. Our model shows state-of-the-art performance for all correct clustering sizes without any discriminative training and with tuning only one convergence parameter.

CCS CONCEPTS

• **Information systems → Entity resolution**;

KEYWORDS

Author Disambiguation; Probabilities; Agglomerative Clustering

ACM Reference Format:
Tobias Backes. 2018. Effective Unsupervised Author Disambiguation with Relative Frequencies. In *JCDL '18: The 18th ACM/IEEE Joint Conference on Digital Libraries, June 3–7, 2018, Fort Worth, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3197026.3197036

1 INTRODUCTION

Documents have authors. This information is almost always available on a document and in the document's metadata. However, it is crucial to distinguish an author name mentioned on a specific document from the author itself. Usually, the author is *referred to* by a string of characters that is given with the document. This concept introduces two types of ambiguity:

(1) *Name synonymy*: One author is referred to by different strings, perhaps due to misspelling, language-specificity, different conventions for name specification, etc.
(2) *Name homonymy*: One string refers to different authors. With the the collection size the chance increases that two authors in the collection have the same name.

In general, both problems have to be addressed simultaneously, i.e. in a constrained clustering setup. This allows to establish that for example *DOE, J* can be *DOE, JW* or *DOE, JH* but not both. In this contribution, we focus on name homonymy and simplify the problem by considering that all initials are always given (assuming that *DOE, J*, *DOE, JW* and *DOE, JH* are different persons). In that case, *Author(ship/name) disambiguation* decides for a set of *author mentions* with the same name, which of them belong to the same author and which do not. This is a clustering problem over author mentions. Each cluster is considered an author. More formally:

- For each collection, there is a set $\mathfrak{N}$ of *names* $name \in \mathfrak{N}$
- For each name *name*, there is a set $\mathfrak{C}$ (also referred to as a *clustering*) of authors $C \in \mathfrak{C}$ (also referred to as a *cluster*) and a set of mentions $x \in X$
- Each author $C \subseteq X$ is a set of *mentions* $x \in C$
- For each mention x, there is a bag of *features* $f \in F(x)$, each with a frequency $\#(f, x)$ of occurrence with x

If a name is not disambiguated, we have no information about the belonging of single mentions. This state can either be expressed by putting all mentions in the same cluster $C = X$ such that $\mathfrak{C} = \{\{x \mid x \in X\}\}$, or by assigning each mention x its own cluster $C = \{x\}$, such that $\mathfrak{C} = \{\{x\} \mid x \in X\}$. The task of author disambiguation is then to suggest a *system* clustering $\mathfrak{C}_{sys}$ that is as close to the correct clustering $\mathfrak{C}_{cor}$ as possible. In the training/tuning and evaluation case, we have both $\mathfrak{C}_{sys}$ and $\mathfrak{C}_{cor}$ present and optimize some evaluation score $eval(\mathfrak{C}_{sys}, \mathfrak{C}_{cor})$. We can safely assume that this score measures the similarity between the system- and the correct clustering. In practice, we disambiguate one name at a time. This is referred to as *blocking* [15], where each name defines one *block*, that is a separate problem set.

Many different approaches have been proposed to solve the problem of author disambiguation. Although it is often not indicated as clearly, most of these approaches can be viewed within the framework and terminology described above. Despite the large number of proposed methods, we find that a straightforward, simple and sound, easy-to-implement – yet well performing – baseline is

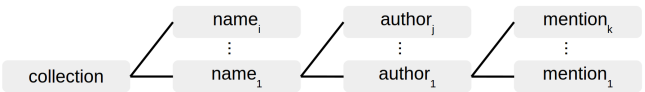


Figure 1: Author disambiguation problem structure

JCDL'18, June 3–7, 2018, Fort Worth, TX, USA

ACM ISBN 978-1-4503-5178-2/18/06…$15.00
https://doi.org/10.1145/3197026.3197036

still lacking. Existing solutions are either complicated to understand and implement, require training steps or are based on a number of rather arbitrary decisions. Normally, performance is not compared against the most primitive conceivable baseline [9]. Therefore, in this work, we elaborate on the following research questions:

RQ1 Can we deploy well-behaved, straightforward probabilities to establish a conceptionally simple, fast and reliable method for author name disambiguation that shows state-of-the-art performance?

RQ2 How difficult is the problem of author name disambiguation in general and separately for different problem sizes? How good are the most primitive baselines and how does our approach compare to them?

We structure the rest of the paper as follows. In section 2, we review the related work. In section 3, we describe the probabilistic model that we designed as a proof-of-concept for RQ1. In section 4, we explain the evaluation setup, including simple baselines and the separation of different problem sizes. This allows us to test our approach regarding RQ1 and gather insights towards the more general RQ2. We summarize the most prominent findings regarding these two research questions in section 5.

2 RELATED WORK

In this section, we give a brief overview over the most cited literature on author disambiguation that relates to our approach:

The Problem of Author Disambiguation. Ferreira et al. [2] give an overview of author disambiguation, distinguish author grouping (*synonymy*) vs. author assignment (*homonymy*) and different types of features for the latter. Smalheiser and Torvik [11] go into more detail, but to some extend also concerning their own project; Harzing [6] analyses the top 1% cited academics from Thompson Reuters Essential Science Indicators and contributes some details of the ambiguity problem in different languages. She shows that the extent of ambiguity has a direct influence on the scientific performance indicators measuring a scientist's academic output. Strotmann and Zhao [13] investigate the application of author disambiguation to citation networks. They show that one of the reasons for researchers being top-ranked is in fact a lack of author name disambiguation. Kramer, Momeni and Mayr [7] give an overview of the quantity of author gold annotation in the Web of Science. They conclude that the Web of Science researcher-ID used in this paper is a good source of author identity information, in contrast to other sources. Milojevic [9] analyzes the accuracy of name blocks built by considering all available initials of an author (and his last name). On the small sample used, this accuracy introduces a very high baseline.

Probabilistic Approaches, Topic Models. Han et al. [5] present a probabilistic Naive Bayes mixture model to disambiguate a small set of highly ambiguous author names. They compare to K-means and find that their model performs significantly better. However, they use a very long product, which can lead to extreme swings and the EM algorithm, a computationally and conceptionally rather complex method. Tang et al. [14] suggest a general probabilistic model based on Markov Random Fields to exploit a variable set of features for author disambiguation. They evaluate their approach on a small set of highly ambiguous names and compare the results to more basic clustering techniques, claiming significant improvements. With ~89% F1, their results are indeed impressive, but considerable effort is required to understand and implement the method. Torvik et al. [16] present a probabilistic model that compares pairs of authors based on various features such as terms, affiliations or venues. Torvik and Smalheiser [15] apply an improved version of this previous approach to the Medline data set and reach ~95% F1 for the one name reported. Their work is one of the most cited state-of-the-art methods in this field. Generally speaking, their experience allows the group to hard-code a lot of knowledge into their methods, but not all decisions can be easily comprehended by others. Although they use some probabilities in their formulas, it seems the final similarities are not normalized. Song et al. [12] present a probabilistic graphical topic model for hierarchical clustering of author names and compare their approach to other standard clustering approaches. They find their model performs better than for example DBSCAN. Although the good performance of ~90% F1 seems to justify the additional complexity, like Han et al., they introduce long products and inference steps on a 'detour' via topic assignments.

The Web of Science, Training Techniques. Like us, Gurney et al. [4] work on the Web of Science and use a very similar set of features. They deploy another group's method for clustering and the Taminoto coefficient for similarity. They report F1>90% for a number of names. Their approach is the one most similar to our proposal, but neither does it use feature-specificity nor return probabilities. Levin et al. [8] present a semi-supervised approach to author disambiguation on the Web of Science data set. The evaluation of their approach shows solid performance (but under 90% F1) and proves that the method scales to very large collections. As they do pairwise classification of author mentions, they make the problem harder than it needs to be. Overall, their work is very detailed but also quite difficult to reproduce. Ferreira et al. [3] present a semi-supervised classifier evaluated on DBLP and BDBComp data sets. Their model outperformed the other methods compared. Depending on supervision and rules, their model introduces a number of (dataset-specific) parameters. Recently, this group has updated their method to allow for incremental disambiguation by comparing not author mentions but mentions to clusters of mentions. [10] This is very practical but more difficult. In another paper focusing on training aspects, Culotta et al. [1] present a special similarity function and combine error-driven sampling with learning-to-rank. They also give an overview of other training approaches. They find that their approach can be beneficial in terms of performance. Similar to Ferreira et al., this approach requires training and uses specific similarity measures for different feature types.

3 METHOD DESCRIPTION

We intent to define a simple, yet effective probabilistic method for author disambiguation. Using the blocking paradigm, our method disambiguates one name at a time. It clusters all mentions of that name based on features extracted from the collection. We note that for each mention x, there is exactly one document $d(x)$ in which this mention appears. However, for each document, there can be multiple mentions that appear on it.

3.1 Features

Features $F(x)$ assigned to a mention x can be extracted (1) from $d(x)$ in general or (2) specifically for x on d. In the first case, the features $F(x)$ are the same as the features $F(\dot{x})$ of a mention $\dot{x}$ that appears on d as well. In the following, we will distinguish different *feature-types*, some of which are *unspecific* for x and some of which are *specific*. We train and test our approach on the Web of Science, as it is the major source of scientific documents annotated with actual author IDs (*researcher-ID*). The following information is used:

(1) *Terms* $F_{term}(x)$: All words considered relevant in the document and their frequency of occurrence in $d(x)$
(2) *Affiliations* $F_{aff}(x)$): All affiliations given for x on $d(x)$ – usually just one
(3) *Categories* $F_{cat}(x)$: All categories assigned to $d(x)$, where we consider categories to be relatively general terms picked from a relatively small vocabulary / thesaurus. The frequency of one category for a document d is usually 1
(4) *Keywords* $F_{key}(x)$: All keywords assigned to $d(x)$, where we consider keywords to be relatively specific terms that are picked from a relatively large vocabulary or only with respect to the current document. The frequency of one keyword for a document d is usually 1
(5) *Coauthornames* $F_{co}(x)$: All names of the coauthors of x on $d(x)$. Unless more than one author of d have the same name, the frequencies are 1
(6) *Refauthornames* $F_{ref}(x)$: All author names from documents d' in the collection such that d references d'. Frequencies larger than 1 will occur often, for example if multiple documents by the same author are referenced
(7) *Emails* $F_{email}(x)$: All email addresses given for x on $d(x)$ – usually just a single email address with a frequency of 1
(8) *Years* $F_{year}(x)$: A bag of years f given for $d(x)$. Like Levin et al. [8], we model $\#(f, x)$ as a Gaussian with the publication year of x as mean. This models temporal proximity.

While there are many details related to the question of which and how features are extracted and normalized, the focus of our research was not to investigate specific features but to develop a method that can provide satisfying results independent of the exact set of features and feature-types.

3.2 Agglomerative clustering

We apply a method of agglomerative clustering as we consider it the most straightforward approach. This means that we start with the initial state where each mention x is in its own cluster $C = \{x\}$. Then, pairs $(C, \dot{C})$ of clusters are merged. If no stopping criterion is applied, this will ultimately result in a state where all mentions are in the same cluster $C = X$. For this reason, we need to compute the score $score(C, \dot{C})$ of a pair $(C, \dot{C})$ of clusters to be merged. Furthermore, we deploy a *quality limit* l, that tells us whether the score can be considered good or not. In our approach, $score(C, \dot{C})$ is not dependent on the score of any other pair of clusters. Neither is the quality limit. This means that in each iteration of the clustering process, we merge all pairs $(C, \dot{C})$, such that (1) $\neg\exists\ddot{C} \in \mathfrak{C} : score(C, \ddot{C}) > score(C, \dot{C}) \wedge \neg\exists\ddot{C} \in \mathfrak{C} : score(\ddot{C}, \dot{C}) > score(C, \dot{C})$ and at the same time, (2) $score(C, \dot{C}) > l$. In other words, we evaluate all disjoint pairs $(C, \dot{C}) \in \mathfrak{C} \times \mathfrak{C}$; for each of these pairs, we check whether (1) and (2) hold true. If yes, the pair is saved for merging. See algorithm 1 for a more formal description. At the end of each iteration, all saved pairs are merged. A new system clustering is obtained and the next iteration begins. This process converges if no pairs are saved for merging. For evaluation purposes, we can continue to merge with moves that are below the limit, but we will elaborate on this in the experiments section.

Algorithm 1: Our agglomerative clustering (no evaluation)

Input: *name* with X and C as well as $N, \#(f), \lambda, \epsilon, l$
while $|C| > 1$ **do**
  $score \leftarrow NULL$;
  foreach $(C, \dot{C}) \in C \times C$ **do**
    $score(C, \dot{C}) \leftarrow \sum_{ftype} \lambda_{ftype} \cdot p_{ftype}(C|\dot{C})$;
  $merges \leftarrow \{\}$;
  foreach $(C, \dot{C}) \in C \times C$ **do**
    if $\neg\exists\ddot{C} \in C : score(C, \ddot{C}) > score(C, \dot{C})$ ***and*** $\neg\exists\ddot{C} \in C : score(\ddot{C}, \dot{C}) > score(C, \dot{C})$ ***and*** $score(C, \dot{C}) > l$ **then**
      $merges \leftarrow merges \cup (C, \dot{C})$;
  if $|merges| = 0$ **then** break ;
  else
    foreach $(C, \dot{C}) \in merges$ **do**
      $merge((C, \dot{C}))$;

3.3 Probabilistic similarity

The main contribution of our approach is the similarity used to define $score(C, \dot{C})$. In the following, we will define and explain the probability that inspires the score. We say that the score of a pair of clusters can be seen as their *joint probability* $p(C, \dot{C}) = p(\dot{C}, C) = p(C|\dot{C}) \cdot p(\dot{C})$. Remember $\#(f, x)$ denotes the frequency of f in the set $F(x)$ of all features in x ($F = \bigcup_{x \in X} F(x)$). Consider $\#(f, x) = 0$ if $f \notin F(x)$. We define:

$$p(C|\dot{C}) = \sum_{(x,\dot{x}) \in C \times \dot{C}} p(x|\dot{x}) \cdot \frac{\#(\dot{x})}{\#(\dot{C})} \qquad p(C) = \sum_{x \in C} p(x)$$

$$p(x|\dot{x}) = \sum_{f \in F} \frac{\#(f, x) \cdot \#(f, \dot{x})}{\#(f) \cdot \#(\dot{x})} \qquad p(x) = \frac{\#(x)}{\#(\centerdot)}$$

$$\#(C) = \sum_{x \in C} \#(x) \qquad \#(x) = \sum_{f \in F} \#(f, x)$$

$$\#(f) = \sum_{x \in X} \#(f, x) \qquad \#(\centerdot) = \sum_{x \in X} \#(x)$$

To prevent division by zero, we apply *add*ϵ smoothing. This modifies $p(C|\dot{C})$, $p(\dot{x})$ and $p(x|\dot{x})$:

$$p(C|\dot{C}) = \sum_{(x,\dot{x})} p(x|\dot{x}) \cdot \frac{\#(\dot{x}) + \epsilon}{\#(\dot{C}) + |C| \cdot \epsilon} \qquad p(x) = \frac{\#(x) + \epsilon}{\#(\centerdot) + |X| \cdot \epsilon}$$

$$p(x|\dot{x}) = \frac{1}{\#(\dot{x}) + \epsilon} \cdot \left(\left(\sum_{f \in F} \frac{\#(f, x) \cdot \#(f, \dot{x})}{\#(f)} \right) + \frac{\epsilon}{|X|} \right)$$

3.4 Variations

So far, we have considered the clustering to be fully enclosed in the current name block. While still treating each name as a separate clustering problem, we can say that X is not only the set of mentions for the current name, but for the entire collection. This increases $\#(f)$, $\#(\bullet)$ and obviously $|X|$. We find that the performance is considerably better if this approach is taken, which can be explained with less sparsity for $\#(f)$. However, this poses the question, what the 'collection' is that we take these counts from. Basically, we are simply looking for a realistic distribution of the frequency of features f independent of the mention they occur with. In an application scenario, this distribution can be taken from the data to be clustered itself. As $\#(f)$, $\#(\bullet)$ and $|X|$ are available even for the data we cluster, in our evaluation scenario, we obtain them from the union of the training and testing portion.

In our preferred variant, we only use $p(C|\dot{C})$ in $score(C, \dot{C})$. This conditional probability does in fact perform much better than the joint probability. Intuitively, $p(C)$ favours large clusters for merging, which does not make sense as it introduces a tendency that reinforces itself. It does not matter that $p(C|\dot{C}) \neq p(\dot{C}|C)$ as merging is symmetric and the clustering procedure we described above will simply use $max(p(C|\dot{C}), p(\dot{C}|C))$ unless there is a third cluster that matches even better (or the limit is not met). In this case, $p(\dot{x})$ is never applied, so that we do not need $\#(\bullet)$. Collecting $\#(f)$ over the entire collection, we then only need the number N of mentions in the collection. Therefore, X still denotes the set of mentions for one single name block. Furthermore, we tested a variant where we replace the sum of products with a maximum of products:

$$\tilde{p}(C|\dot{C}) = \max_{(x,\dot{x}) \in C \times \dot{C}} p(x|\dot{x}) \cdot \frac{\#(\dot{x})}{\#(\dot{C})}$$

Obviously, the value $\tilde{p}$ inspired by single-link clustering is not a probability anymore. This variant performed slightly different than the conditional probability.

3.5 Feature-type weights

All the probabilities shown above are obtained *separately* for each feature-type. Consider that each probability p should actually be denoted p_{ftype} where $ftype$ is either *term*, *aff*, *cat*, *key*, *co*, *ref*, *email* or *year*. For better readability, we drop the subscript where it is not necessary. We perform a simple linear combination with feature-type weights λ to obtain the final score:

$$score(C, \dot{C}) = \sum_{ftype} \lambda_{ftype} \cdot p_{ftype}(C|\dot{C})$$

Ideally, we would like to avoid all training so as to keep our method as simple as possible. Still, to allow for comparison, weights λ are trained on the training portion of our data. We sample pairs (x, C) and $(x, \dot{C})$ such that $x \in C \wedge x \notin \dot{C} \wedge |C| = |\dot{C}| \wedge C \cup \dot{C} \subseteq X$, where X is the set of mentions for a *single* name. All possible values for $|C| = |\dot{C}|$ are considered in order to create a more or less realistic binary classification scenario, where we are asked to assign x to a correct cluster C or an incorrect cluster $\dot{C}$. The classifier (*logistic regression* performed well) receives probabilities $p_{ftype}(x|C)$ and $p_{ftype}(x|\dot{C})$ for each $ftype$ together with the class 'correct' or 'incorrect'. It then learns feature-type weights λ_{ftype} in order to optimize the classification outcome. While this is not the same scenario as the one that the weights are finally applied in, we hope that nevertheless, we gather some insight into the importance of single feature-types.

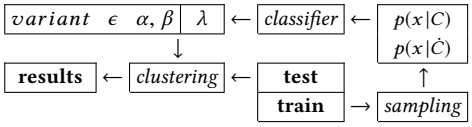


Figure 2: Setup for clustering, evaluation and training λ

3.6 Convergence

Above, we have introduced a quality limit l on the scores for moves during clustering. In order to account for different problem sizes (and corresponding smaller probabilities), we define l as follows:

$$l = \alpha + |X| \cdot \beta$$

where $|X|$ is the number of mentions for the current name. Fortunately, when they are normalized such that $\sum_{ftype} \lambda_{ftype} = 1$, l is relatively independent of λ. Another pleasant finding was that it is sufficient to tune one parameter depending on whether we use the sum-of-products or the maximum-of-products variant. While we optimize both parameters, results were best if in the first case $\beta = 0$ and in the second $\alpha = 0$.

3.7 Implementational Details

Our approach as presented in previous sections can be implemented in an efficient and conceptionally simple way by means of matrix multiplication. In a first step, we calculate the matrix $\overline{p(x|\dot{x})}$ containing values $p(x|\dot{x})$ for all pairs $(x, \dot{x}) \in X \times X$. This is based on the $|X| \times (|F| + 1)$ count matrix $\overline{\#(x, f)}$, the feature count vector $\overrightarrow{\#(f)}$, and the mention count vector $\overrightarrow{\#(x)}$. For smoothing, $\overline{\#(x, f)}$ is extended by one column $\langle \epsilon \dots \epsilon \rangle^T$.

$$\overline{p(x|\dot{x})} = \overline{p(x|f)} \cdot \overline{p(f|\dot{x})} \qquad \overline{p(x|f)} = \frac{\overline{\#(x,f)}}{\overrightarrow{\#(f)}^T}$$

$$\overline{p(f|\dot{x})} = \frac{\overline{\#(x,f)}}{\overrightarrow{\#(x)}} \qquad \forall i : \overline{\#(x,f)}_{i,|F|+1} = \epsilon$$

Here, $\cdot$ denotes matrix multiplication (*matrix- or dot product*). During each iteration, we calculate the $|\mathfrak{C}_{sys}| \times |\mathfrak{C}_{sys}|$ matrix $\overline{p(C|\dot{C})}$ from $\overline{p(x|\dot{x})}$ and the current clustering as a $|X| \times |\mathfrak{C}_{sys}|$ matrix $\overline{\mathfrak{C}_{sys}}$:

$$\overline{p(C|\dot{C})} = \overline{\mathfrak{C}_{sys}}^T \cdot \left(\overline{p(x|\dot{x})} \cdot \overline{p(\dot{x}|C)} \right) \qquad \overrightarrow{\#(C)} = \overrightarrow{\#(x)}^T \cdot \overline{\mathfrak{C}_{sys}}$$

$$\overline{p(\dot{x}|C)} = \left(\left(\overrightarrow{\#(x)} + \epsilon \right) \cdot \frac{1}{\overrightarrow{\#(C)} + |\overline{\mathfrak{C}_{sys}}| \epsilon} \right) \circ \overline{\mathfrak{C}_{sys}}$$

Here, $\circ$ denotes component-wise multiplication (*Hadamard product*), ensuring $p(x|C) = 0$ if $x \notin C$. For the *max* variant, all sums in the matrix products are replaced by a maximum function.

4 EXPERIMENTAL EVALUATION

In the following we describe the experimental setup used to train, tune and test our approach. This is also depicted in figure 2.

4.1 Data

We use the *Web of Science* (*WoS*) collection as a source of metadata and annotated authorship information. In a preprocessing step, we extract features for the feature-types mentioned in the previous section. We normalize author names as LASTNAME, INITS, where I, N, I, T, S are all the initials for each first name of the author mention. Thereby, on the one hand, we make the problem harder as we drop the full name information (increasing the problem size, i.e. *John Doe* = *Jack Doe*) and on the other we make it simpler as we treat mentions of the same author where his first names are given with varying completeness as separate problems (reducing the problem size, i.e. *John Doe* ≠ *John W. Doe*). We extract terms and their frequency from the title and abstract of the metadata. Title terms are weighted three times higher than abstract terms. Some stop words are omitted and all words are lower-cased. Furthermore some basic lemmatization from the *Natural Language Toolkit* (*NLTK*) is applied. Affiliations are already normalized in the WoS. Categories and keywords are taken as they are in the WoS. Co- and referenced author names are normalized as described above. Emails are not normalized, but we plan to lowercase them in the future. From the publication date, we only pick the year.

After extracting features for the entire WoS with more than 100 million documents, we create a database containing the features related to each single mention with a researcher-ID. For each name, we use all the mentions that are given a researcher-ID and we use all the authors that contain at least one such mention. As stated earlier, we consider each researcher-ID a distinct author. Names are ordered randomly and separated into training and testing portions. We use 25% names for training.

4.2 Test setup

In contrast to work by most other groups, we are specifically interested in evaluating the performance of our model in relation to the problem size. If this has been done in the literature, it is usually regarding the number $|X|$ of mentions with the same name (i.e. in Gurney et al. [4]). However, we sort our name blocks by the correct number of clusters that would need to be detected in order to achieve perfect results (the size $|\mathfrak{C}_{cor}|$ of the cluster*ing*). We compare a maximum of 1000 names for all $|\mathfrak{C}_{cor}| \in \{1..10\}$. Of this selection, 25% are used for training. Clustering sizes are distributed according to Zipf's law. For the sizes 1 to 4, more than 1000 names are available. See table 1 for exact number of names for each clustering size. We evaluate our approach for each size separately. Doing so, we are basically balancing our data in an artificial way. We find this is necessary, as the Zipf distribution of $|\mathfrak{C}|$ leads to a (usually unobserved) preference of models that create very few clusters. Furthermore, in order to view the actual behavior of our method, we carefully monitor the development of precision, recall and F1 measure with each iteration of the clustering process. We also monitor how the clustering would have continued if there were no limit l on the score of possible merges. So for each iteration in the clustering process, we record the following information:

(1) **Precision** of current system clustering
(2) **Recall** of current system clustering
(3) Current **number of clusters** in system clustering
(4) Whether the current iteration is before **convergence**

$\lvert\mathfrak{C}\rvert$	1	2	3	4	5	6	7	8	9	10	$\sum$
train	*255*	*250*	*250*	*250*	*215*	*139*	*80*	*65*	*43*	*35*	*1582*
test	*767*	*751*	*750*	*750*	*645*	*418*	*242*	*195*	*131*	*106*	*4749*
train+test	*1022*	*1001*	*1000*	*1000*	*860*	*557*	*322*	*260*	*174*	*141*	*6331*
all	*229653*	*14108*	*3630*	*1657*	*860*	*557*	*322*	*260*	*174*	*141*	*251362*
% used	*0.45*	*7.10*	*27.55*	*60.35*	*100*	*100*	*100*	*100*	*100*	*100*	*2.52*

Table 1: The number of names found or used with a correct clustering size $|\mathfrak{C}_{corr}| \in \{1..10\}$ in the WoS data

During the clustering process, we continuously apply the quality limit l to merges. Once an iteration is reached where no possible merge exceeds the limit, the following iterations are only hypothetical and all possible merges are applied if and only if no merge exceeds the limit. Note that this means that even during hypothetical iterations, there might be a limit-based selection of merges. As there is always at least one possible merge, all clusters are finally merged into one and we can evaluate the development of precision and recall as well as the point of convergence. This is particularly interesting if one has certain preferences towards precision or recall and would like to find a good stopping point for the clustering process. Figure 6 shows how we plot the recorded information.

4.3 Evaluation measures

In order to evaluate the performance of our approach, we use two popular evaluation measures for clustering: (1) *pairwise F1* (*pairF1*) and (2) *bCube*. Both measures define *precision* (*P*) and *recall* (*R*) when comparing two clusterings $\mathfrak{C}_{sys}$ and $\mathfrak{C}_{cor}$. F1 is defined as usual as $2 \cdot \frac{P \cdot R}{P+R}$. Except for one aspect, we use the definition by Levin et al. (2012) [8]:

$$pairs(\mathfrak{C}) = \bigcup_{C \in \mathfrak{C}} \{\{x, \dot{x}\} \mid x, \dot{x} \in C \wedge x \neq \dot{x}\}$$

$$P_{pairF1} = \frac{pairs(\mathfrak{C}_{cor}) \cap pairs(\mathfrak{C}_{sys})}{pairs(\mathfrak{C}_{sys})}$$

$$R_{pairF1} = \frac{pairs(\mathfrak{C}_{cor}) \cap pairs(\mathfrak{C}_{sys})}{pairs(\mathfrak{C}_{cor})}$$

$$C_{sys}(x) = C \in \mathfrak{C}_{sys} : x \in C$$

$$P_{bCube} = \frac{1}{|X|} \cdot \sum_{x \in X} \frac{|C_{sys}(x) \cap C_{cor}(x)|}{|C_{sys}(x)|}$$

$$R_{bCube} = \frac{1}{|X|} \cdot \sum_{x \in X} \frac{|C_{sys}(x) \cap C_{cor}(x)|}{|C_{cor}(x)|}$$

We note that the above shown P_{pairF1} and R_{pairF1} are not defined if in the first case $\mathfrak{C}_{sys}$ and in the second $\mathfrak{C}_{cor}$ are $\{\{\{x\} \mid x \in X\}\}$. Therefore we modify $pairs(\mathfrak{C}) = \bigcup_{C \in \mathfrak{C}} \{\{x, \dot{x}\} \mid x, \dot{x} \in C\}$. This increases the values for pairF1 slightly.

One important question with regard to these evaluation measures is on which subset of the problem they are applied. It is understood from the above formula, that there is a distinct precision and recall value for each clustering problem, that is for each name. However, one could also consider the pairs to be taken over the entire test data:

$$pairs_{cor}(\mathfrak{N}) = \bigcup_{name \in \mathfrak{N}} \bigcup_{C \in \mathfrak{C}_{cor}^{name}} \{\{x, \dot{x}\} \mid x, \dot{x} \in C \wedge x \neq \dot{x}\}$$

In that case one would calculate one value of precision and recall over all correct and incorrect pairs in the test data. From these two values, F1 could be calculated. If we calculate precision and recall for each name separately, we have to average over the results that we get for each single name. This weights each name equally (independent of the number $|X|$ of mentions with that name). We can then calculate F1 from the average precision and average recall over all names. We use this approach to obtain a final score for each correct number $|\mathfrak{C}|$ of clusters. We do not report a final score over all $|\mathfrak{C}_{cor}|$ as we aim to establish a more precise evaluation for the different cases that are possible. However, in table 1 we report the number of names that were found in the WoS data for each $|\mathfrak{C}_{cor}|$, from which one can approximate the performance over the whole collection. As the performance of our approach does not vary to any relevant extend between using *pairF1* or *bCube*, we show only plots for *bCube*.

4.4 Experiments

In our experiments, we use the setup and the measures described above in combination with different parameters, hyper-parameters and variants. Our model has the following variants:

(1) *within* / *overall*: $\#(f)$ only within one name or over all
(2) *pc_on* / *pc_off*: using $p(C, \dot{C})$ or using $p(C|\dot{C})$
(3) *prob* / *max*: sum-of-products or maximum-of-products

As indicated earlier, for most variants, results show that only one of the two options was worth further investigation. We choose the following setup based on first experiments on the training data: (1) *overall* and (2) *pc_off*. The difference between (3) *prob* and *max* was not as clear. We decided that the *max* variant is worth further investigation but focused mostly on the *prob* variant as it could be implemented to run much faster and first results also gave a better F1 score. Our model has the following (hyper-) parameters:

(1) Smoothing hyper-parameter ϵ
(2) Feature-type weights λ
(3) Convergence parameters α and β

So far, we have tried only one very small smoothing parameter $\epsilon = .0001$ to avoid division by zero. We train the feature-type weights over the union of training portions for all clustering sizes and approximate the results as shown in table 2. This table also shows all other feature-type weights examined. Observing stopping size vs. correct size, we tuned the limit parameters in a manual grid-search on the training data and found that a good choice is to set $\alpha = .0005$ for the *max* variant and $\beta = .000075$ for the *prob* variant. Our plots include a histogram of the clustering sizes $|\mathfrak{C}_{cor}|$ where the system clustering converged. All plots are given for the training data as they have been part of the parameter tuning process.

4.5 Results

As mentioned above, we record a number of measurements during the clustering process in order to understand the behavior of our method and the effect of different versions, stopping limits and feature-type weightings. Table 3 shows the final results with tuned parameters on the test portion. In addition to that, we present our measurements in two types of plots. Referring to figure 6 as an example, we briefly explain how to read the more comprehensive type:

term	.15	.12	.125	1	0	0	.143	0	.2
aff	.2	.03	.125	0	0	.143	.143	0	0
cat	.18	.1	.125	0	0	.143	.143	0	.2
key	.03	.2	.125	0	⋱ 0	.143	⋱ .143	0	.2
co	.2	.02	.125	0	0	.143	.143	.5	.2
ref	.12	.15	.125	0	0	.143	.143	.5	0
email	.1	.18	.125	0	0	.143	.143	0	0
year	.02	.2	.125	0	1	.143	0	0	.2
	train	***opp.***	***unif.***	***select***		***leaving-one-out***		***author***	***doc.***

Table 2: Feature-type weights considered in our experiments

The y-axis displays the interval $[0, 1]$, onto which precision, recall and F1 are mapped. The x-axis gives the number of clusters in a clustering iteration. For a single block, clustering starts somewhere on the left of the plot with very low recall and maximal precision. As the process continues to merge clusters, recall increases and precision decreases. Over all blocks, we see the same development, but averaged for each problem size (standart deviation shown). In a perfect method, precision would remain constant until the correct number of clusters, shown as a solid vertical line, is reached. In figure 6, we see that for the *max* variant, F1 peaks exactly at this point, while a bit later for the *prob* variant. For both variants, the empirical stopping size (our method does not know the correct number of clusters) for $|\mathfrak{C}| = 5$ peaks where F1 is maximal. The offset of this empirical distribution is tuned with the stopping parameters α and β. We can see in figure 7 that there is still room for improvement of the stopping limit l, as for $|\mathfrak{C}| = 10$ our method generally stops later than it should.

As preliminary tests on the training data clearly favour one combination of variants and the stopping parameters are easily tuned for to these two options, the most interesting comparison is between different feature-type weightings. As a first choice, trained weights from the classifier are used and give satisfying results (see fig. 3). The *max* variant performs worse in terms of F1, but precision is higher (see fig. 4). Detailed plots (fig. 6) of the clustering process suggest that the *prob* variant can sometimes gain relatively much recall in early stages of the clustering, while the *max* variant has particularly regular behavior with smaller deviations from the mean. It is also interesting to see that the *max* variant is able to gain the maximal F1 at the correct number of clusters, while the *prob* variant achieves higher values of F1 (due to better recall) but peaks at a clustering size larger than the correct one. The average maximum recall (*'max rec.'*) at perfect precision (which is independent of the stopping parameters) shown in figures 3 and 4 is much higher with the *prob* variant, which supports the notion that the *prob* variant has its strengths in a high recall. On the other hand, the average maximum precision (*'max prec.'*) at perfect recall is slightly higher for the *max* variant, suggesting that in general it can keep precision higher until the end. We also show precision and F1 for the baseline of putting all mentions into the same cluster (*'base prec.'* and *'base f1'*). Obviously, recall is 1 here.

The trained feature-type weighting is contrasted with a uniform weighting of feature-types. Results are almost identical as can be seen in figure 10, suggesting that equal weighting of all feature-types is a good choice. However, this does not mean that feature-type weighting has no influence on the performance. In

		1	2	3	4	5	6	7	8	9	10	$\lvert\mathfrak{C}\rvert$					$\lvert\mathfrak{C}\rvert$	1	2	3	4	5	6	7	8	9	10	
bCube		*100*	*95*	*95*	*96*	*97*	*96*	*94*	*96*	*95*	*94*	**P**	trained	$\alpha = 0$	**prob**	$\beta = .000075$	trained	**P**	*100*	*95*	*95*	*96*	*96*	*96*	*94*	*95*	*94*	*93*	**pairF1**
		98	***93***	***93***	***92***	***91***	***90***	***91***	***91***	***90***	***89***	**F**						**F**	***99***	***94***	***93***	***92***	***92***	***91***	***92***	***92***	***91***	*90*	
		97	*91*	*90*	*88*	*86*	*85*	*87*	*87*	*86*	*84*	**R**						**R**	*97*	*92*	*92*	*89*	*88*	*87*	*90*	*89*	*88*	*87*	
		100	*95*	*95*	*97*	*97*	*97*	*95*	*96*	*96*	*95*	**P**	uniform				uniform	**P**	*100*	*95*	*95*	*96*	*97*	*96*	*94*	*96*	*94*	*94*	
		98	***93***	***93***	***92***	*90*	***90***	*90*	*90*	*89*	***89***	**F**						**F**	***99***	***94***	***93***	***92***	*91*	*91*	*91*	*91*	*90*	***91***	
		97	*91*	*90*	*87*	*84*	*84*	*85*	*85*	*83*	*84*	**R**						**R**	*97*	*92*	*91*	*89*	*86*	*85*	*88*	*87*	*85*	*87*	
		100	*96*	*96*	*97*	*96*	*95*	*92*	*92*	*91*	*88*	**P**	trained	$\alpha = .0005$	**max**	$\beta = 0$	trained	**P**	*100*	*96*	*96*	*97*	*96*	*95*	*92*	*90*	*89*	*87*	
		96	*92*	*91*	*91*	*90*	***90***	*89*	*89*	*89*	*88*	**F**						**F**	*97*	*92*	*92*	*92*	*90*	*90*	*89*	*88*	*88*	*88*	
		93	*88*	*87*	*86*	*84*	*85*	*87*	*86*	*87*	*88*	**R**						**R**	*94*	*89*	*88*	*87*	*84*	*85*	*87*	*86*	*86*	*89*	
		100	*96*	*96*	*97*	*96*	*95*	*92*	*92*	*91*	*88*	**P**	uniform				uniform	**P**	*100*	*96*	*96*	*97*	*96*	*94*	*91*	*91*	*89*	*87*	
		96	*92*	*91*	*91*	*89*	***90***	*89*	*89*	*88*	*88*	**F**						**F**	*97*	*92*	*92*	*91*	*89*	*89*	*89*	*85*	*87*	*88*	
		92	*88*	*87*	*86*	*83*	*85*	*86*	*86*	*85*	*88*	**R**						**R**	*93*	*89*	*87*	*86*	*83*	*85*	*87*	*88*	*84*	*89*	

Table 3: Results of the tuned method on the test portion, using bCube and pairF1 measure

order to investigate the influence of feature-type weighting, we test an 'opposed' weighting where the feature-type with the previously smallest weight is assigned the previously biggest weight, the feature-type with the previously biggest weight is assigned the previously smallest weight, and so on (see table 2). Results are clearly worse (see fig. 5), which shows that feature-type weighting is not completely irrelevant (in the sense that there are also counterproductive weightings). Furthermore, we use feature-type weighting to investigate the effect of single feature-types by on the one hand leaving them out (setting their weight to zero) and on the other using them exclusively (setting all others to zero). This analysis shows that most feature-types can be dropped (if they are the only one to be dropped) with the exception of co-authors (see fig. 11) and ref-authors. It is therefore interesting to see the results if only these two feature-types are used (see fig. 12). Performance is solid. The most important coreference indicators are co-author names, which might even be used alone (see fig. 9) . In a last experiment, we also test a weighting that selects only basic features of the document $d(x)$ (no features associated with the author mention x itself and no referenced authors). Results (fig. 13) are acceptable: F1 is not as good as before, but precision is high.

4.6 Discussion

A direct comparison of our experimental results to those obtained by other researchers is not possible. In fact such a comparison would almost certainly not be fair as evaluation results depend heavily on so many different factors that reproducibility is close to impossible under normal circumstances:

(1) *Dataset*: size, distribution of authors, version of data set, domain, availability of features, completeness of author name specification, hand-selected?, quality and amount of gold-annotation
(2) *Blocking scheme*: average size of blocks, unassigned names, overlapping blocks?
(3) *Evaluation measure*: general choice of measure, micro/macro average?, recall only inside block?, pair- or element-wise comparison?, counting pairs of equal mentions?, evaluated for different problem sizes?

It is therefore more than desirable to have a benchmark dataset with a framework that distinguishes clearly between *data*, *annotation*, *blocking*, *disambiguation* and *evaluation*. As a such benchmark is not available at a realistic scale like the Web of Science, we promote the paradigm to evaluate different problem sizes (number of clusters that need to be found) separately. This makes our results relatively robust against changes to many (not all) of the above mentioned factors. Unfortunately, this kind of evaluation can not be referenced from other publications that have worked on the Web of Science. Therefore, we base the assessment of the AD method proposed in this work on the fact that we were able to achieve results of more than 90% F1 for problem sizes of at least ten authors, which we find promising. Closely related work by Levin et al. [8] or Gurney et al. [4] does not report much higher values even though they do not separate the problems sizes so rigorously – which generally improves the results. However, we must note that Levin et al. cluster blocks of surname and first initial, while we use *all* initials. Therefore, they work with much larger problem sizes, which is computationally challenging and leads to more realistic estimation of recall, even if measured only within blocks. We hope that reporting results for individual problem sizes minimizes these distortions in the comparison.

5 CONCLUSIONS

In the following, we briefly summarize the main findings of our research: The best variant is using $p(C|\dot{C})$ instead of $p(C, \dot{C})$ with $\#(f)$ over the whole collection. Using maximum-of-products instead of sum-of-products constitutes a serious alternative. There are some hints that it might be even more precise (*'max prec.'* is slightly higher). Certainly, it runs significantly slower in our implementation, which is why we have not yet fully investigated potential gains of fine-tuning this variant. When tuning an appropriate stopping parameter, our method can deliver state-of-the-art results although being conceptionally simple. Even though feature-type weights learned in the classification scenario are quite heterogeneous, when applied in the clustering application, they do not perform better than uniformly distributed weights. We view this as a benefit of our model, as its score works well independent of any training. We hypothesize that the probabilities filter out unspecific features, thereby implicitly controlling feature weighting without any discriminative training. The stopping criterion needs to be tuned on some training set, but it is only a single variable per variant that needs to be fitted. This quality limit does remarkably well at finding an appropriate number of clusters to converge. Leaving out

one feature-type at a time in the clustering score, we find that co-author names are the most important features. Apart from this, our method's performance is not dependent on the presence of specific feature-types. For example, a weighting where only the co-author names and the referenced author names are used performs well. Recording the results for each system clustering size $|\mathfrak{C}_{sys}|$ separately allows to precisely monitor the behavior of the clustering process. The plots created in this work also allow tuning precision vs. recall. This might be particularly interesting for digital libraries, as they might prefer precision over recall. Separate evaluation for each correct clustering size $|\mathfrak{C}_{cor}|$ shows how high the baseline of putting all mentions in a single cluster is for the frequent cases of $|\mathfrak{C}_{cor}| = 1$ or $|\mathfrak{C}_{cor}| = 2$. We conclude that, with larger problems not separated, an approach could easily be considered satisfying although only approximating this primitive baseline.

ACKNOWLEDGMENTS

This work was supported by the EU's Horizon 2020 programme under grant agreement H2020-693092, the MOVING project.

REFERENCES

[1] Aron Culotta, Pallika Kanani, Robert Hall, Michael Wick, and Andrew McCallum. 2007. Author disambiguation using error-driven machine learning with a ranking loss function. In *Sixth International Workshop on Information Integration on the Web (IIWeb-07), Vancouver, Canada.*

[2] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. 2012. A brief survey of automatic methods for author name disambiguation. *Acm Sigmod Record* 41, 2 (2012), 15–26.

[3] Anderson A Ferreira, Adriano Veloso, Marcos André Gonçalves, and Alberto HF Laender. 2010. Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries.* ACM, 39–48.

[4] Thomas Gurney, Edwin Horlings, and Peter Van Den Besselaar. 2012. Author disambiguation using multi-aspect similarity indicators. *Scientometrics* 91, 2 (2012), 435–449.

[5] Hui Han, Wei Xu, Hongyuan Zha, and C Lee Giles. 2005. A hierarchical naive Bayes mixture model for name disambiguation in author citations. In *Proceedings of the 2005 ACM symposium on Applied computing.* ACM, 1065–1069.

[6] Anne-Wil Harzing. 2015. Health warning: might contain multiple personalities - the problem of homonyms in Thomson Reuters Essential Science Indicators. *Scientometrics* 105, 3 (2015), 2259–2270.

[7] T. Kramer, F. Momeni, and P. Mayr. 2017. Coverage of Author Identifiers in Web of Science and Scopus. *ArXiv e-prints* (March 2017). arXiv:cs.DL/1703.01319

[8] Michael Levin, Stefan Krawczyk, Steven Bethard, and Dan Jurafsky. 2012. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology* 63, 5 (2012), 1030–1047.

[9] Staša Milojević. 2013. Accuracy of simple, initials-based methods for author name disambiguation. *Journal of Informetrics* 7, 4 (2013), 767–773.

[10] Alan Filipe Santana, Marcos André Gonçalves, Alberto HF Laender, and Anderson A Ferreira. 2017. Incremental author name disambiguation by exploiting domain-specific heuristics. *Journal of the Association for Information Science and Technology* 68, 4 (2017), 931–945.

[11] Neil R Smalheiser and Vetle I Torvik. 2009. Author name disambiguation. *Annual review of information science and technology* 43, 1 (2009), 1–43.

[12] Yang Song, Jian Huang, Isaac G Councill, Jia Li, and C Lee Giles. 2007. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries.* ACM, 342–351.

[13] Andreas Strotmann and Dangzhi Zhao. 2012. Author name disambiguation: What difference does it make in author-based citation analysis? *Journal of the American Society for Information Science and Technology* 63, 9 (2012), 1820–1833.

[14] Jie Tang, Alvis CM Fong, Bo Wang, and Jing Zhang. 2012. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering* 24, 6 (2012), 975–987.

[15] Vetle I Torvik and Neil R Smalheiser. 2009. Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3, 3 (2009), 11.

[16] Vetle I Torvik, Marc Weeber, Don R Swanson, and Neil R Smalheiser. 2005. A probabilistic similarity metric for Medline records: A model for author name disambiguation. *Journal of the American Society for information science and technology* 56, 2 (2005), 140–158.

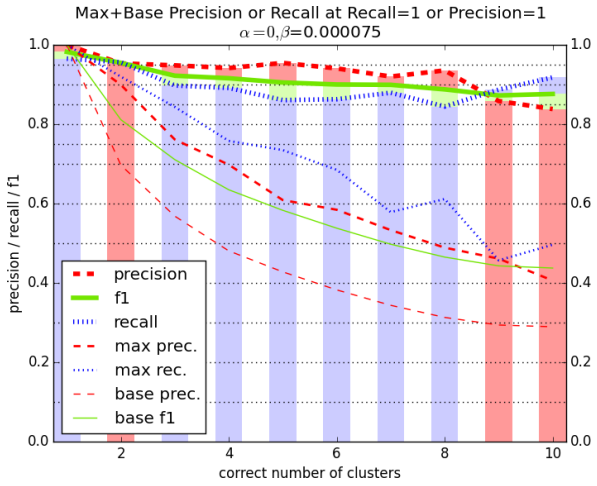


Figure 3: Results for trained weights λ;using *prob* variant

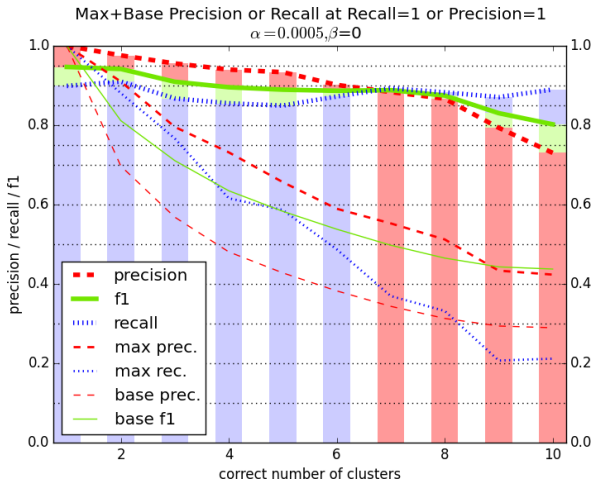


Figure 4: Results for trained weights λ; using *max* variant

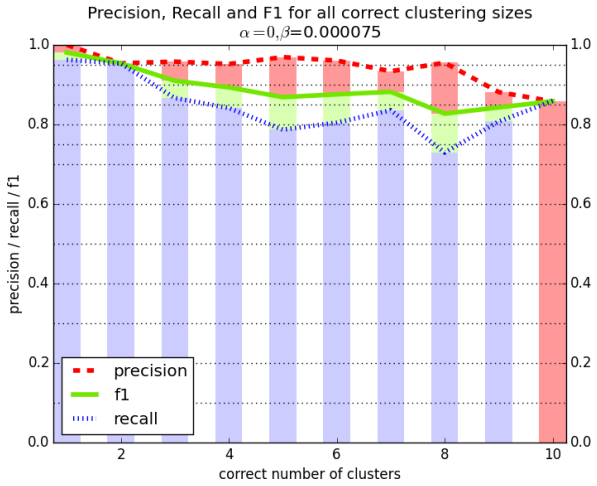


Figure 5: Results for opposed weights (compare figure 3)

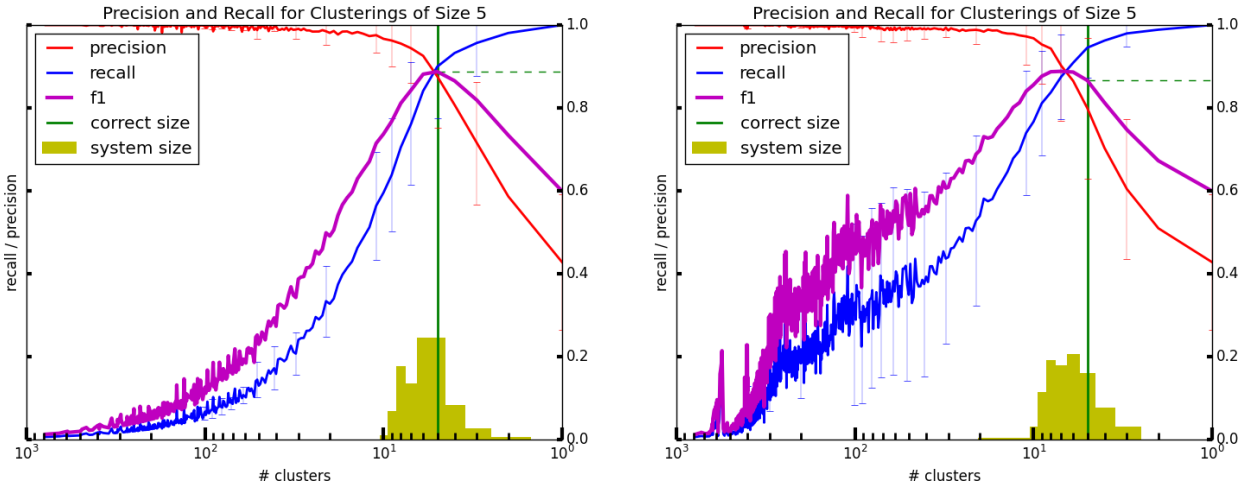


Figure 6: The clustering process visualised for trained weights λ; comparing *max* and *prob* variant; $|\mathfrak{C}| = 5$

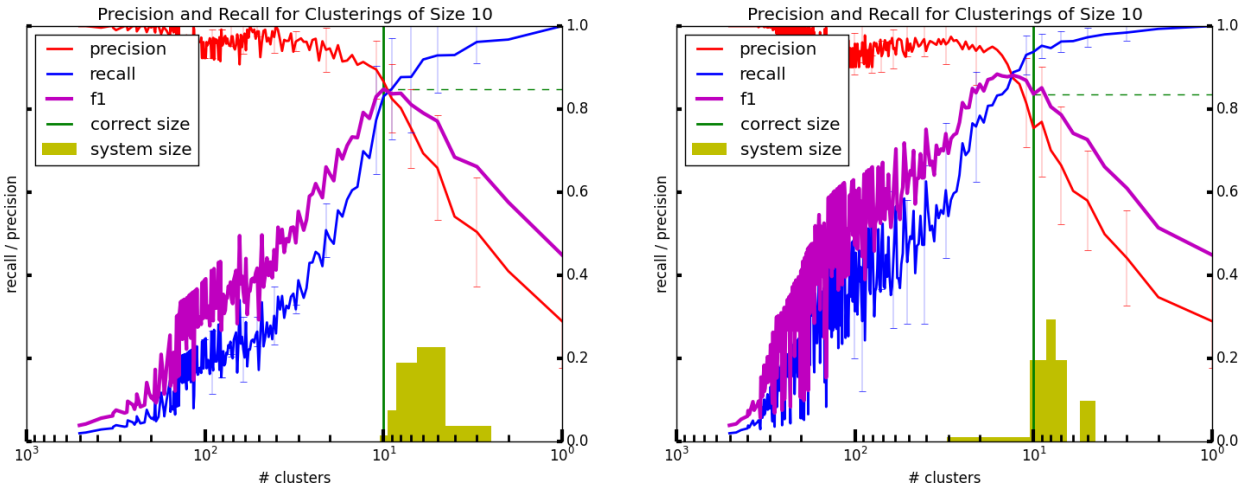


Figure 7: The clustering process visualised for trained weights λ; comparing *max* and *prob* variant; $|\mathfrak{C}| = 10$

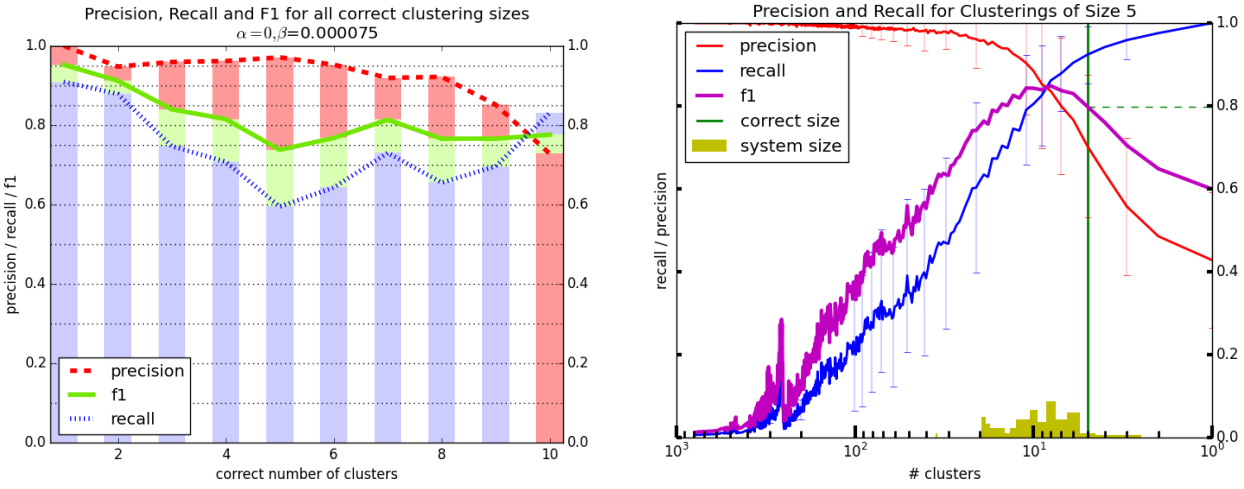
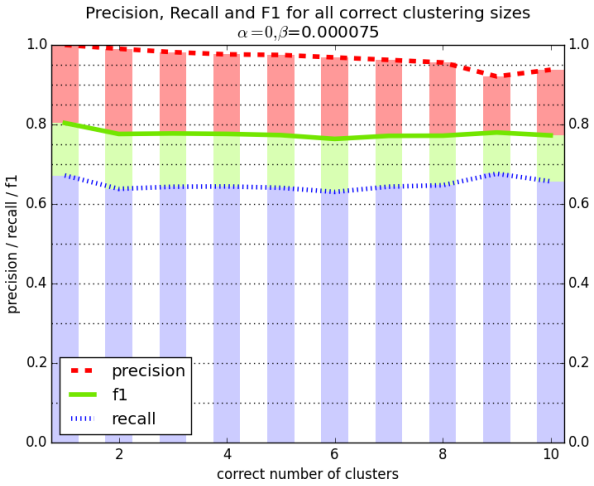


Figure 8: Selecting only terms as features; results and details of the clustering process for $|\mathfrak{C}| = 5$



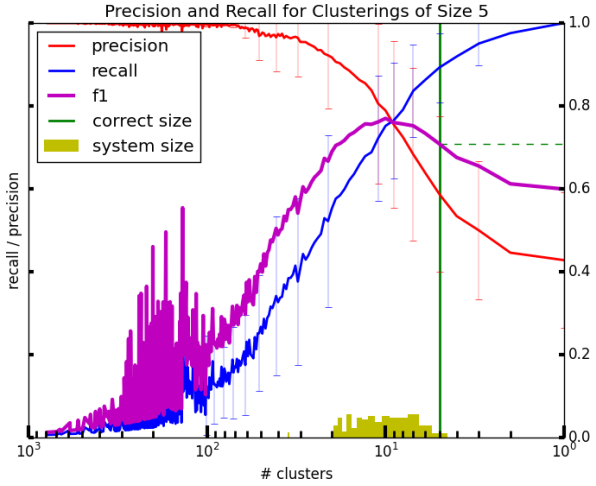


Figure 9: Selecting only co-authors as features; results and details for $|\mathfrak{C}| = 5$

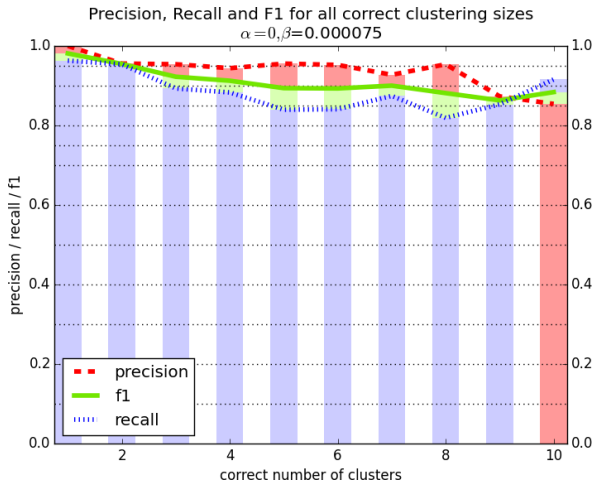


Figure 10: Results for uniform weights (compare figure 3)

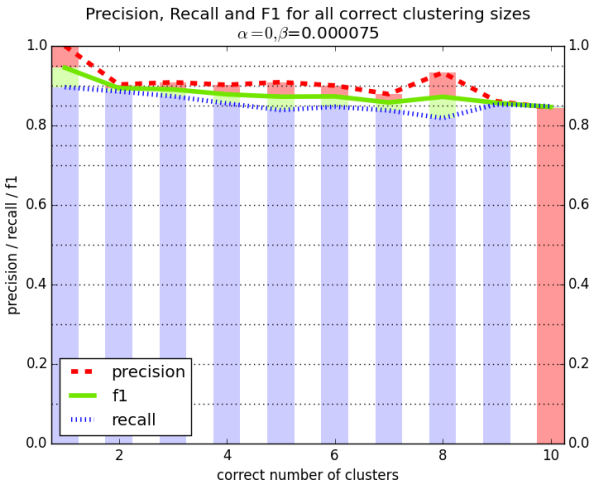


Figure 12: Only co- and referenced authors (compare fig. 3)

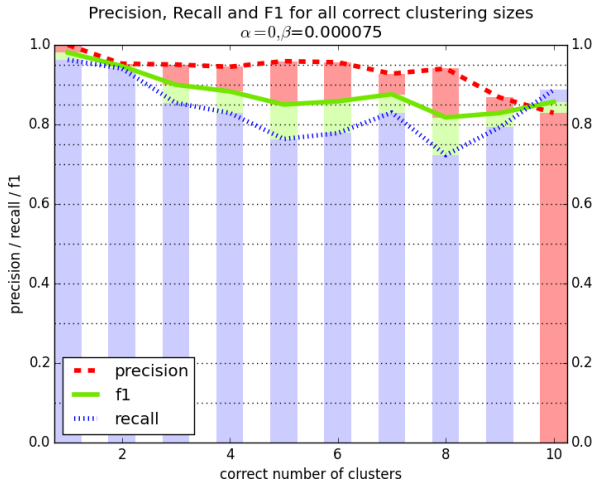


Figure 11: Leaving out co-authors (compare figure 3)

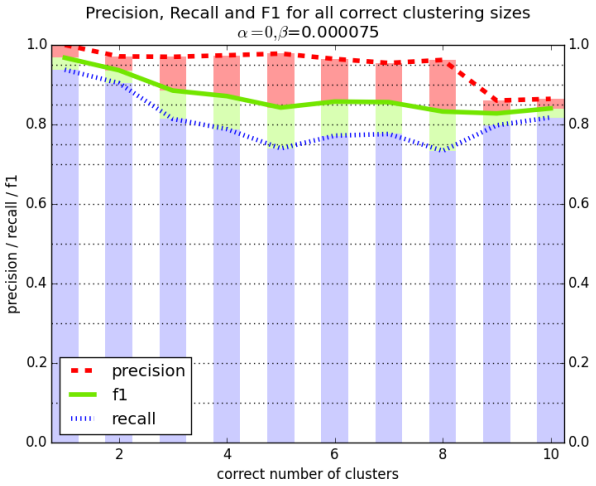


Figure 13: No author-specific features (compare figure 3)