



Norwegian University of
Science and Technology

Cumulative Citation Recommendation

Christian Barth Roligheten

Master of Science in Computer Science

Submission date: June 2018

Supervisor: Kjetil Nørvåg, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Keeping knowledge bases such as Wikipedia up-to-date with the latest information is a difficult task in the information age: Every day thousands of news articles, blog posts, opinions are published on the Internet and if we imagine that just a small fraction of these documents contain new information that would require a knowledge base to be updated, then we need an army of constantly vigilant volunteers to keep track of this stream of information and update knowledge bases as it becomes necessary. Obviously as more more information is generated on the Internet, we need increasingly more volunteers to keep track of it all. It would then be greatly beneficial if we could create automated systems which assist volunteers with integrating new information into knowledge bases.

Cumulative Citation Recommendation (CCR) is the task of assisting knowledge base editors by automatically recommending edits to entity profiles in knowledge bases given a stream of documents. In this thesis we implement a CCR system that allow us to evaluate different learning-to-rank (LTR) based ranking approaches to CCR. Specifically we compare entity-dependent and entity-independent approaches, as well as approaches which use Gradient Boosted Trees and Random Forests as the ranking algorithm. We also evaluate how different features affect the system. Our best approach which uses Gradient Boosted Trees and an entity-dependent approach achieves an F1 measure of 0.5 on the 2014 TREC KBA track, which would places it in second place compared to other participants of this track. Our evaluation of different LTR-based approaches reveal which approaches are most effective for CCR.

Sammendrag

Å holde kunnskapsbaser som Wikipedia oppdatert med ny informasjon er en utfordrende oppgave i informasjonsalderen: Hver dag blir tusenvis av nyhetsartikler, blogginnlegg og meninger publisert på nettet og hvis vi forestiller oss at bare en liten mengde av disse krever at en kunnskapsbase oppdateres, trenger vi en arme av oppvakte redaktører for å holde oversikt over denne strømmen med informasjon og oppdatere kunnskapsbaser når nødvendig. Åpenbart trenger vi en alltid økende mengde redaktører ettersom mer informasjon produseres på nettet over tid. Det ville vært til stor hjelp dersom vi kunne laget systemer som støtter frivillige redaktører med å integrere ny informasjon inn i kunnskapsbaser.

Cumulative Citation Recommendation (CCR) handler om å støtte redaktører i kunnskapsbaser ved å automatisk anbefale endringer til entitetsprofiler i kunnskapsbaser basert på en strøm av dokumenter. I denne avhandlingen bygger vi et CCR system som lar oss evaluere forskjellige learning-to-rank (LTR) baserte metoder til CCR. Spesifikt sammenlikner vi entitetsavhengige og entitetsuavhengige metoder og metoder basert på Gradient Boosted Trees og Random Forests som rangeringsalgoritmer. Vi evaluerer også hvordan forskjellige valg av features påvirker systemet. Vår beste metode oppnår en F1 poengsum på 0.5 som plasserer denne metoden på andreplass i TREC KBA track 2014. Vår evaluering av forskjellige LTR baserte metoder viser også til hvilke metoder som fungerer for CCR i forskjellige situasjoner.

Acknowledgments

I would like to thank my supervisor Professor Kjetil Nørvåg for his guidance and for being supportive and encouraging throughout this project. I would also like to thank Professor Krisztian Balog and Associate Professor Heri Ramampiaro for taking time off their busy schedules to provide me with valuable insight into the field of research covered by this thesis.

Lastly I would like to thank my friends and fellow thesis writers Dag Erik Løvgren and Øystein Aas Eide for fruitful discussions throughout the semester and for giving me a reason to show up at school every day outside of writing this thesis.

Table of Contents

Abstract	i
Sammendrag	ii
Acknowledgments	iii
Table of Contents	vii
List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Motivation	1
1.2 Research questions	4
1.3 Outline	5
2 Preliminaries	7
2.1 Supervised learning	7
2.2 Document processing	10
2.3 Text representations for supervised learning	11
2.4 Evaluation of information retrieval systems	13

3	Related Work	17
3.1	Cumulative citation recommendation	17
3.2	Knowledge base population	19
4	TREC Cumulative Citation Recommendation	21
4.1	The TREC KBA track	21
4.2	Task description and definitions	22
4.3	Document collection and truth data	24
4.4	Evaluation metrics	27
5	Implementation	29
5.1	Architectural overview	29
5.2	Building word vectors for temporally sensitive tasks	30
5.3	The document processing system	31
5.4	The Vital Filtering System	44
6	Experiments & Results	49
6.1	Document collection and truth data	49
6.2	Evaluation methodology	50
6.3	Experimental overview	51
6.4	Results	52
7	Discussion & Conclusion	57
7.1	Discussion of results	57
7.2	Contribution	60
7.3	Conclusion	62
7.4	Future work	63
	Bibliography	65

A Per-topic training data distribution	69
Appendix	69
B Command-line used for evaluating runs	73

List of Tables

4.1	Target entity types for the 2014 CCR task and the number of entities in each category	22
4.2	Subset of fields in a StreamCorpus document which we use in this thesis. Descriptions adapted from streamcorpus.org	24
4.3	Evolution of the TREC KBA StreamCorpus between the 2012 and 2014 KBA track	25
5.1	Statistics of different collections which we combine and use to train our word-embeddings.	31
5.2	Basic features	37
5.3	Temporal features	39
5.4	Variants of document-representation features we implemented	43
5.5	Verb tags from the Penn Treebank English POS tag set and the tense they indicate	44
5.6	Tense and clause features	44
6.1	Variants of document-representation features we implemented	51
6.2	Performance of different feature sets when using an entity-independent approach	52
6.3	Performance of different feature sets and approaches when using an entity-dependent approach	53
6.4	Best F1 and SU score of each participating team in the 2014 TREC KBA track and our best (bold)	56

A.1	Number of training examples for each relevancy level per entity in the training data of the 2014 TREC KBA track (Target entities of the CCR task only)	71
-----	--	----

List of Figures

1.1	Number of articles vs number of active editors (>5 edits) on Wikipedia	2
1.2	Illustration of the process of Cumulative Citation Recommendation	3
2.1	Typical method of splitting truth data into separate training and testing sets	8
2.2	A typical LTR system	9
4.1	Example of StreamCorpus document containing noisy text unrelated to the article (bold)	27
5.1	High-level architecture of our system	30
5.2	Architecture of the document processing system	31
5.3	Partial hypernym tree of the verb win in Wordnet, verbs that indicate vitality in a hypothetical CCR classifier are labeled in green	42
5.4	Architecture of the Vital Filtering System	45
5.5	Architecture of the entity-dependant ranking approach	48
6.1	Graph showing the F1 measure of different feature sets and approaches when using an entity-independent approach	53
6.2	Graph showing the F1 measure of different feature sets and approaches when using an entity-dependent approach	54
6.3	Graph showing the F1 measure of different feature sets for all approaches	55

Introduction

In Section 1.1 we will detail the motivation behind this thesis, we will then use the problems described in the motivation to define research questions which this thesis will attempt to answer in Section 1.2. We will then briefly summarize the contributions of this thesis in Section 7.2 and then outline the structure of this thesis in Section 1.3.

1.1 Motivation

Increasingly in the modern era we are relying more and more on knowledge bases such as Wikipedia as a source of information in our everyday lives, however while modern knowledge bases are convenient for the average person, people often do not think of what goes on in the background in order to make sure these knowledge bases are kept up-to-date and relevant.

For free and open knowledge bases such as Wikipedia, an army of volunteer editors is needed to constantly update articles or create new ones when needed, and while the Internet has created a platform for these editors to easily collaborate, it has also created a problem: As the velocity of information creation on the Internet increases, more work from editors is needed to keep knowledge bases up-to-date. This problem is illustrated in Figure 1.1: While the number of active editors has been falling since around 2007, the number of articles is increasing linearly and shows no sign of slowing down. As time passes and more articles are being created, it will become increasingly difficult for a small number of editors to keep up with the ever increasing amount of new information created both on the Internet and in the real world.

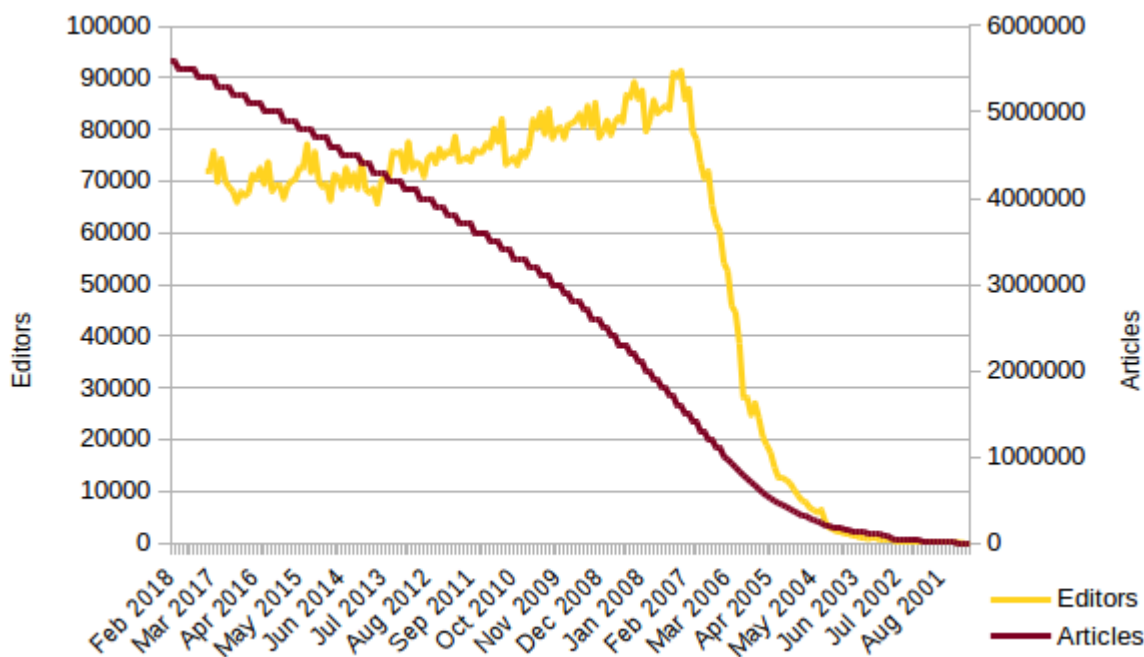



Figure 1.1: Number of articles vs number of active editors (>5 edits) on Wikipedia

We can think of two ways to address the problem of keeping knowledge bases up-to-date: Either the number of editors has to increase proportionally with the number of articles, or the effectiveness of each editor needs to increase such that they can keep up with the increasing complexity. While Computer Science may not be able to address the latter, it may be able to address the former: By creating systems which can effectively notify editors of new important events related to the topics in knowledge bases, then it will be easier for a small number of editors to keep up with changes to increasingly complex knowledge bases.

Cumulative Citation Recommendation (CCR) is the task of assisting knowledge base editors by automatically recommending edits to entity profiles in these knowledge bases given a stream of documents. Figure 1.2 illustrates this process: Given a stream of documents which can include news articles, blog posts, etc. it is the task of CCR systems to automatically process this stream of documents and filter out what is relevant to different entities. The highly relevant filtered documents are then accumulated and presented to knowledge base editors who integrate this new content into existing knowledge base profiles.


This thesis will focus on the problem of building a CCR system that can effectively filter out documents that are relevant to an entity given a document collection containing potentially millions of documents. Our goal when implementing this system is to uncover which methods work when building such a system so that we can guide future research.

Given a stream of document...



Watch: Racing presenter amazingly catches runaway horse
 Entertainment.ie - 9 hours ago
 At The Races presenter Hayley Moore shocked everyone earlier in the week when she managed to catch a loose horse at the Chepstow ...

Filter documents highly relevant to entity



Elon Musk wants to build a transport network underground and ...
 The indy100 - 21. mai 2018
 Elon Musk might not appeal to everyone but there is no denying that he has done some great things in the name of science and progress.

Recommend edits to Knowledge Base profile

Elon Reeve Musk FRS (/ˈɪ.lɒn/; born June 28, 1971) is a South African-born American business magnate, investor^[9] and engineer.^[10] He is the founder, CEO, and lead designer of SpaceX,^[11] co-founder, CEO, and product architect of Tesla, Inc.; and co-founder and CEO of Neuralink. In December 2016, he was ranked 21st on the *Forbes* list of *The World's Most Powerful People*.^[12] As of February 2018, he has a net worth of \$20.8 billion and is listed by *Forbes* as the 53rd-richest person in the world.^[13]

Born in Pretoria, South Africa, Musk taught himself computer programming at the age of 12. He moved to Canada when he was 17 to attend Queen's University. He transferred to the University of Pennsylvania two years later, where he received an economics degree from the Wharton School and a degree in physics from the College of Arts and Sciences. He began a Ph.D. in applied physics and material sciences at Stanford University in 1995 but dropped out after two days to pursue an entrepreneurial career. He subsequently co-founded Zip2, a web software company, which was acquired by Compaq for \$340 million in 1999. Musk then founded X.com, an online payment

Elon Musk
FRS



Musk in 2015

Born Elon Reeve Musk
 June 28, 1971 (age 46)
 Pretoria, South Africa

Residence Bel Air, Los Angeles, California, U.S.^[12]

Figure 1.2: Illustration of the process of Cumulative Citation Recommendation

1.2 Research questions

With the motivation described in Section 1.1 we define our main research question as follows:

RQ1 How can we effectively filter out new, relevant information related to a set of entities given a stream of documents?

This main research question will be answered by implementing a system to filter relevant information from a stream of documents, and evaluating if it does indeed effectively filter new relevant information related to a set of predefined entities. Implementing such a system however is no small task, and many different approaches have been proposed in existing research. We will follow earlier work on CCR such as Balog et al. (2012) and Jiang et al. (2014) which find that supervised learning approaches using learning-to-rank (LTR) achieve some of the best results for the problem.

When implementing the CCR system we will also evaluate different approaches to creating such a system. In existing research on CCR a popular machine-learning algorithm has been the Random Forests algorithm, we want to see if we can get better results using Gradient Boosted Trees. We also want to evaluate if entity-dependent approaches achieve better results than entity-independent approaches for the system we implement. Finally we will look at how the choice of features affect the performance of our system for these different implementations. For this we define three secondary research questions that prompt us to look at different ways of creating a LTR based system and comparing them with the purpose of guiding future work on this problem:

RQ2 How does the Random Forest algorithm compare to Gradient Boosted Trees when used for ranking?

RQ3 How does entity-dependent approaches compare to entity-independent approaches for ranking?

RQ4 What features are effective when used in conjunction with the different approaches?

These research questions will be answered by implementing a system which can run with both the Random Forests and Gradient Boosted Trees algorithm, while for both of these approaches also having the ability to switch between entity-dependent and entity-independent approaches. By comparing these approaches using the appropriate metrics with different features we can make recommendations as to what works and what does not work for CCR.

1.3 Outline

Chapter 2 - Preliminaries: Gives an overview of the theoretical concepts that need to be understood in order to properly implement our approach.

Chapter 3 - Related Work: Presents related work in both CCR and related fields

Chapter 4 - TREC Cumulative Citation Recommendation: Gives an overview of the TREC KBA track and related CCR task which we will implement and evaluate our implementation against

Chapter 5 - Implementation: Details our implementation of a CCR system build in the context of the TREC KBA track.

Chapter 6 - Experiments & Results: Explains our approach to evaluating the system we implemented and the results of this evaluation.

Chapter 6 - Discussion & Conclusion: Presents a discussion of the results of the evaluation and our answers to the research questions. Lastly we present our conclusion to this thesis and future work.

Preliminaries

In order to properly approach the task of creating a system which solves the task of cumulative citation recommendation we need to understand the preliminary theory underlying the creation of such a system. As we stated earlier there are many ways of creating CCR systems, and covering the theory behind all such approaches would be beyond the scope of this thesis. In this chapter we will therefore give an overview of the established theoretical concepts which we will base our implementation and evaluation on.

2.1 Supervised learning

Supervised learning is the problem of taking a set of labeled examples and using these to make predictions on unseen data points (Mohri et al., 2012). The specific type of prediction one wishes to make can vary depending on the task. Common types of problems for supervised learning are classification, regression and ranking problems which are all differentiated by the type of unseen data we wish to predict. The methods used to make predictions in all of these methods vary and it would be outside the scope of this thesis to cover all of them. In this section we will give a brief overview of the generic topics of supervised learning as well as more specific topics of special interest to this thesis.

2.1.1 Features

When using supervised learning we have stated that we use labeled examples to make prediction on unseen data points. These examples that are used to make prediction have to be structured in a way that allow a supervised learning algorithm to utilize them to make these predictions. In supervised learning the collection of data-points for any given training example are often called “features” or “attributes” of the given example. It is normal for any given example to be associated with many different features which make up a “feature vector”. The goal of any supervised learning algorithm is to build models which take as input a feature vector and output the correct prediction for that vector.

2.1.2 Training and testing set

After using a given algorithm to create a model it is customary that we want to evaluate the performance of this model which requires us to have some labeled examples that we can compare the models predictions to. A naive approach to evaluating the model would be to simply have the model make predictions on the labeled examples it used to create the model, and see how accurate it is on recreating the labels. This is however not an appropriate way to evaluating a model because it does not evaluate if the model is only good at predicting labels it has seen before and does not test if the model generalizes well to predicting unknown feature vectors, something that is called overfitting.

Because of this it is normal to split the collection of all available labeled examples (truth data) into two separate sets which are called the training and testing set as shown in Figure 2.1. The training set of labeled examples is made available to the supervised learning algorithm while the testing set is kept hidden. When it comes to evaluating a given model the labeled examples from the testing set are used to evaluate the models predictions. Since the model did not have access to these labels for examples in the testing set, the performance of the model when predicting these labels give a good measure of the model’s generalized performance.

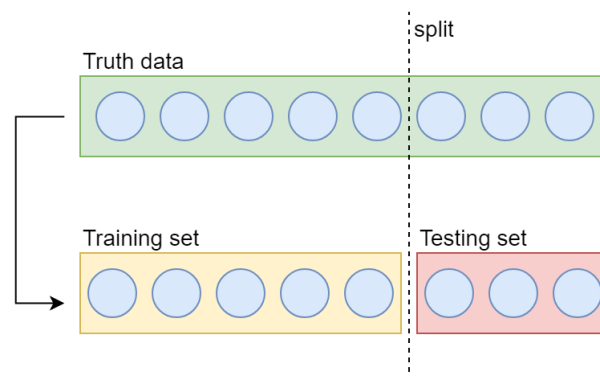


Figure 2.1: Typical method of splitting truth data into separate training and testing sets

2.1.3 Learning-To-Rank

Ranking is one of the central problems of Information Retrieval (Liu, 2011). According to Baeza-Yates A. et al. (1999) the primary problem of any IR system is to predict which documents a user will find useful or not given a query. In order to predict which documents are relevant and which are irrelevant, a ranking model is used to rank a set of documents according to their relevancy. Producing this ranking model is one of the primary problems of IR.

Learning-To-Rank (LTR) is the task of automatically constructing a ranking model using training data (Liu, 2011). Figure 2.2 illustrates the process of creating and using an LTR model for ranking. As we can see from this figure, learning-to-rank is a supervised-learning approach to ranking: A set of labeled training data are used to train a ranking model, this model can then be used to make prediction on unlabeled testing data.

Since learning-to-rank attempts to solve the ranking problem in IR, it is also natural that it takes into account the query-based nature of ranking in IR: Since documents are only ranked in relation to a query, rankings of documents from different queries are not comparable. Because of this labeled training data in LTR systems is grouped by their respective query. When a ranking model is trained, model fitness is evaluated only in query groups. In the same fashion when ranking unknown documents are ranked relative to other documents for the same query.

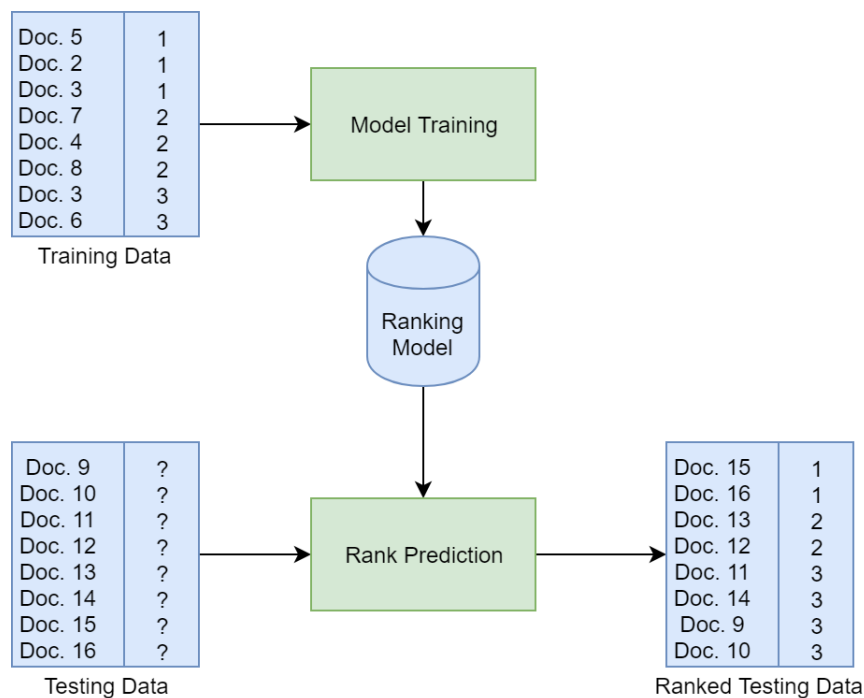


Figure 2.2: A typical LTR system

LTR is not a single algorithm and encompasses many different approaches, in Liu (2011) they summarize that LTR there are primarily three types of LTR algorithms that can be distinguished by how they predict rankings for a query given a group of documents:

Pointwise approaches take as input only single documents and attempts to predict their relevancy to a query without taking into account rankings of other documents for the same query. Because the pointwise approach does not take into account the query-based nature of ranking in IR, it is considered in Liu (2011) to have certain limitations.

Pairwise approaches take as input pairs of documents and attempts to predict which of the two documents are more relevant to the given query. According to Liu (2011) the ranking then boils down to a classification problem where the task is to reduce the number of miss-classified pairs of documents. Again this approach does not fully exploit the fact that ranking problems in IR are query-focused, and in a similar fashion to the pointwise approach it does not fully exploit the query-based nature of ranking problems in IR.

Listwise approaches take as input the whole set of document associated with a query. The task of ranking them becomes to produce a permutation of the document set associated with a query that best represents the true ranking of the documents associated with a query. The listwise approach is according to Liu (2011) the one that best represents the task of ranking in IR given that it uses whole sets of documents associated with a query, something neither the pairwise or pointwise approach does.

2.2 Document processing

Document pre-processing is an important part of any information retrieval system and is especially important for supervised learning systems which use features based on document-representations: When textual content in documents are used to make predictions, then the amount of noise in the textual content can negatively affect the accuracy of the predictions made. The effects of pre-processing on document content on classification performance has been investigated in Uysal et al. (2014) where they find that the method of document pre-processing can affect the performance of a classification system and that careful considerations of pre-processing method can significantly improve classifier performance.

In this section we will quickly introduce the methods of document pre-processing that are relevant for this thesis.

Tokenization

Tokenization is the process of separating a stream of characters into tokens. A token in this case is usually a single word or term that existed in the original stream of characters. Extracting these terms can be tricky because it is not always apparent where the boundary of a term is. An example of this given in Manning et al. (2008) is tokenizing the stream of characters “O’Neill”. Here it is not apparent if we should consider this as a single token or if we should split it into two tokens in the form of “O” and “Neill”. Resolving such conflicts is an important consideration we need to make when tokenizing since the best choice may be dependent what you are planning to use the tokens for later.

Stopword removal

Stopword removal is the process of removing common words that provide little meaning to the underlying text from the textual content of the document. Stop word removal is commonly performed after tokenization as tokenization makes it easy to identify certain stop words from a given list of terms to be removed. This list of words to be removed is called the stop word list, and can either be computed by removing terms that appear very frequently in the document collection (Lo et al., 2005), or retrieved from existing stop words lists.

Whether stop word removal has a positive impact on supervised learning problems such as text classification has not been proved and in Manning et al. (2008) they state that a general trend has been to use smaller lists or no lists at all. In more specialized domains such as sentiment analysis it has also been shown that using classical stop word lists can negatively impact system performance (Saif et al., 2014).

Term normalization

Term normalization is the process of transforming similar terms in such a way that they become equivalent. A popular example of term normalization is case-folding which involves lower-casing all character such that terms such as “Book” and “book” become equivalent terms. Another form of term normalization is called lemmatization which involves reducing a word down to its base form. An example of lemmatization is the transformation of verb “Running” to its base form “Run”. Term normalization can be beneficial in that it can drastically reduce the vocabulary size in a document collection, which is especially true for case-folding. However term normalization can also normalize terms that represent different things: Case-folding “Bush” as in “George Bush” to “bush” makes it harder to distinguish that the term refers to a surname and not a bush.

2.3 Text representations for supervised learning

One challenge of applying supervised learning to textual content is how to best represent text in a way that a machine learning algorithm can understand and effectively utilize. As a general rule for a machine learning algorithm to properly discover relationships between text we need to preserve as much of the meaning and context of each word in a text as possible. However we also want to limit the amount of irrelevant information since this will increase the complexity of the model. Finding a way to represent text, and further whole documents with these requirements in mind is crucial to properly apply machine learning to text related tasks.

2.3.1 The classic vector-space model

One of the earliest and simplest ways of representing text is the classic vector-space model (Salton et al., 1975). The vector-space model represents documents in a vector-space where each dimension corresponds to a unique term in the vocabulary. A document can then be represented as a vector in this model where each term has an associated weight which is determined by the frequency of the term occurrence in the document. Many term weighting schemes are used for the vector-space model, the simplest being the binary weighting scheme where the term weight is 1 if the document contains the term and 0 otherwise. The TF-IDF weighting scheme is also commonly used where the TF-IDF weight of the term in relation to the document and collection is used as a weight.

The classic vector-space model can be used to represent documents in supervised learning tasks: By considering each dimension in the vector-space as an individual feature fed to a supervised learning algorithm it is possible to use this model for supervised learning tasks (Baeza-Yates A. et al., 1999).

While the classic vector-space model is widely used in IR tasks it is not without its flaws. When using the vector-space model as a way to represent a given document as a feature-vector there are some issues that have to be dealt with:

Dimensionality and sparsity Since each dimension only corresponds to a single term and a vocabulary can consist of millions of unique terms, the size of the vectors in this model can become very large, in addition any single document is likely to only contain a fraction of the terms, meaning the vector representing a document will be very sparse. This can lead to problems for many machine learning algorithms because the number of features required to represent all terms can introduce unwanted complexity which can lead to overfitting (Meng et al., 2011). While there are approaches to mitigate this problem such as feature selection it is still a core weakness of the vector-space model.

Loss of word meaning and context When a piece of text is converted into a vector as in the vector-space model, a significant amount of semantic and contextual information about the original text is lost. Firstly since each term has a fixed position in the vector, the position of terms in the original text is lost, which gives rise to the vector-space model often being called the “bag-of-words model”. Additionally the vector does not encode semantic relationships such as some terms being semantically related. For example when using the vector-space model, the terms “run” and “sprint” are as related as the terms “run” and “window” or any other term for that matter.

2.3.2 Distributed representations

While the traditional vector-space model represent each word in a single dimension in the vector space, distributed representations represent words as points in a vector-space (Bengio et al., 2003). Relatively recently approaches to generate distributed vectors for words which maintain semantic relationships have gained popularity in NLP related tasks. These include neural-network based approaches such as the skip-gram approach described in Mikolov et al. (2013) as well as count-based approaches such as the GloVe model described in Pennington et al. (2014). As mentioned these approaches aim to preserve semantic relationships between words.

Up until this point we have only discussed distributed vectors of words, which while useful when we work on tasks involving single words, are not so useful when working on sentences or whole documents. A popular approach to this is based on simply aggregating the word-vectors in a text using the mean, max or min function and considering this as a representation of the text as a whole (De Boom et al., 2016). In this thesis we will focus mostly on aggregating word-vectors using the mean function, which can be described as follows.

$$D_{mean} = \frac{\sum_{i=1}^{|D|} v_i}{|D|} \quad (2.1)$$

Where D is a given document, v_i is the word-vector of the i th word and $|D|$ is the number of words in D . The resulting vector D_{mean} can be considered a representation of the underlying document with a distributed representation based on word-embeddings. This representation has the benefit of having fixed dimensionality, ie. the vector dimensionality is equal to the dimensionality of the underlying word-vector no matter how large our vocabulary grows. From the distributional hypothesis we also retain the semantic relationship between documents with semantically similar words because the underlying words will be close to each other in the vector-space. This does not mean that the model is without its flaws: Since we just take the mean of each word-vector, we lose the original context of each word in the document, ie. a document consisting of the same words in random order will have the same representation.

2.4 Evaluation of information retrieval systems

In this section we will describe the evaluation metrics that are relevant for Cumulative Citation Recommendation.

2.4.1 Precision and Recall

Precision and recall are two widely used measure of test accuracy in information retrieval and form the standard evaluation measure for many such systems (Baeza-Yates A. et al., 1999). Precision can be thought of as a measure of a test's "accuracy", ie. the ratio of retrieved documents that actually are relevant and documents retrieved. Recall on the other hand is a measure of how many of the relevant documents that exists that were retrieved by the system or algorithm. In a more formal way we can describe precision in the following way: For a given query Q let A be the set documents the algorithm retrieves a relevant, let R be the set of all the relevant documents to the query Q . The precision of the algorithm for query Q is then as follows.

$$precision = \frac{|R \cap A|}{|A|} \quad (2.2)$$

Recall can be described formally in the same manner as follows.

$$recall = \frac{|R \cap A|}{|R|} \quad (2.3)$$

2.4.2 F-Measure

When comparing retrieval algorithms it is useful to be able to summarize both the precision and recall of the algorithm as a single value (Baeza-Yates A. et al., 1999). Taking the harmonic mean of the precision and recall is one way to summarize these values into a single value, and it is then called the F-Measure or F1-Measure. It can be calculated as follows.

$$F\text{-Measure} = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} \quad (2.4)$$

2.4.3 Scaled Utility

The scaled utility metric is a linear utility measure that is based on rewarding relevant retrieved documents and punishing non-relevant retrieved documents (Robertson et al., 2002). The utility function can be described as follows.

$$T11U = 2 * TP - FP \quad (2.5)$$

Where TP is the number of true positives retrieved by the system and FP is the number of false positives retrieved. Once we have the utility score of the system it is normalized by the maximum possible score a system can achieve, which is calculated as follows.

$$T11NU = \frac{T11U}{2 * (TP + FN)} \quad (2.6)$$

Where FN is number of false negatives. Finally the scaled utility is retrieved by adding the concept of minimal normalized utility to the measure. The minimal normalized utility ($MinNU$) is the maximum negative utility a user will tolerate before the user stops looking for documents. Generally many values for $MinNU$ can be used but normally this value is set to $MinNU = -0.5$ (Robertson et al., 2002). Now we can calculate the scaled utility as follows.

$$T11SU = \frac{\max(T11NU, MinNU) - MinNU}{1 - MinNU} \quad (2.7)$$

The scaled utility incorporates the idea that a user will stop searching for more documents when the utility reaches $MinNU$, hence the utility cannot fall below $MinNU$ even if the actual utility of the search $T11NU$ falls below this value (Robertson et al., 2002).

2.4.4 Statistical significance testing in IR

One central problem in information retrieval is the comparison of the performance of two different systems: Given two systems A and B, how can we determine if one performs significantly better than the other? In Smucker et al. (2007) they present different methods of testing the significance of differences in performance between two IR systems: In TREC related problems data is often separated into N topics and the evaluation metric is calculated separately for each of these topics. Given two systems A and B we then have N paired samples of scores for these two systems, the methods tested in Smucker et al. (2007) use various statistical tests on these N paired scores to calculate a p-value. They come to the conclusion the randomization test, bootstrap shift method test, and Student's t-test produce comparable significance values. In this thesis we will focus on Student's t-test because it is relatively simpler compared to the two other aforementioned methods.

In Sakai (2014) a good guideline for performing a two-sided t-test on paired topic scores for two systems is presented: Given that we have two systems A and B with N paired topic scores where a_i and b_i is the respective system score for topic i , then with μ_A and μ_B being the sample mean across topics for the two respective systems, we want to test the null hypothesis $H_0 : \mu_A = \mu_B$.

In the setting of a two-sided paired Student's t-test we reject H_0 if the test statistic (t_0) is larger than the two-sided critical t value $t(\phi, \alpha)$ where ϕ is the degrees of freedom which will be $N - 1$ and α is the significance criterion. Calculating t_0 can be done as follows:

$$t_0 = \frac{\bar{d}}{\sqrt{V/N}} = \sqrt{N} \frac{\bar{d}}{\sqrt{V}} \quad (2.8)$$

Where $\bar{d} = \sum_{i=0}^N d_i/N$ is the sample mean of the score differences for each topic ($d_i = a_i - b_i$), and $V = \sum_{i=0}^N (d_i - \bar{d})^2 / (N - 1)$ is the unbiased population variance.

Sakai (2014) also states that the p-value should be presented alongside the results of the significance test. The p-value is simply $P(|T| > t_0)$ which can be found using statistical software or tables.

Related Work

In this chapter we review existing approaches and research that is related to the CCR problem. First in Section 3.1 we review relevant related work on the main topic of this thesis. In Section 3.2 we review work that is related to CCR in the field of knowledge base population (KBP).

3.1 Cumulative citation recommendation

Research on cumulative citation recommendation has been largely dominated by the TREC KBA track which ran in 2012, 2013 and 2014. In all of these tracks a variety of approaches have been tried to the CCR task, however since we will be looking at supervised approaches in this thesis, we will look primarily at related work in CCR which uses supervised approaches.

Classification vs ranking approaches

In Balog et al. (2012) K. Balog, et al. summarizes that supervised approaches to CCR can be primarily split into classification- and ranking-based approaches. Ranking-based approaches attempt to rank candidate documents relative to each other such that a knowledge-base editor can easily filter out those that are most relevant (Efron et al., 2012). Classification approaches however such as those presented in Berendsen et al. (2012) attempt to first make a binary decision of which documents are vital and then use the confidence score of the classifier to rank documents relative to each other. Multi-step classification approaches have also been proposed in Balog Krisztian et al. (2013) where they perform multiple stages of binary classification to make more detailed classifications of documents.

Choice of machine-learning algorithm

When using either classification- or ranking-based approaches many different machine-learning algorithms have been used: In Balog Krisztian et al. (2013) they compare approaches based on random forests and the J48 algorithms and discover that generally the random forest algorithm is preferable, similar experiments were performed in Balog et al. (2012) where they also find that for a learning-to-rank based approach the random forest algorithm beats other ranking algorithms with the same feature set. More recent classification approaches such as the one presented in Cano et al. (2014) have attempted to apply randomized tree ensembles to generally good results, while Reinanda et al. (2016) used gradient boosted decision trees. A classification approach that uses a joint deep neural network approach has also been proposed in Ma et al. (2017).

Entity-dependent vs entity-independent approaches

In the first 2012 TREC KBA track most supervised-learning approaches trained a single global ranking or classification model for all entities, which effectively means that their model needs to handle the generalized properties of all entities as opposed to those specific to each entity. In the 2014 TREC KBA track the number of training instances per entity was selected such that enough training instances exist per entity to allow experimentation with entity-dependent approaches which train a single model per entity as opposed to a single global model. Examples of entity-dependent approaches can be found in Wang, Zhang, et al. (2014) where they train both an entity-dependent and entity-independent model and find that their entity-dependent model performs slightly better than the entity-independent one. Another notable entity-dependent approach is Jiang et al. (2014) where they use an entity-dependent ranking approach and achieve significantly better results than all other participants in the 2014 TREC KBA track.

Feature engineering

Feature engineering is an important aspect of building supervised approaches to CCR. In Balog Krisztian et al., 2013 they propose four categories into which they group common features used for CCR. These categories are *document features* which rely only on the document content and its metadata, *entity features* which depend on data about the target entity, *document-entity features* which capture relationships between a document and target entity, and finally *temporal features* which quantify if some important event relevant to the entity is taking place at a given time.

In general the most interesting developments on feature engineering for CCR have been on either temporal or document-entity features. The document and entity feature categories are usually based on representing existing metadata about documents and entities as features and it is therefore limits to how these features can be applied to CCR, for this reason we have not seen much development of these types of features in later CCR approaches.

For temporal features burst detection on various time-series data is often used to detect if something has happened to an entity, and is usually performed on the time-series data looking N hours into the past from when the document was timestamped. Different sources of this time-series data have been used: In Balog Krisztian et al. (2013) they use both number of mentions of the entity in the document collection and Wikipedia page-view data, while Wang, Liao, et al. (2015) used Google Trends as a source of this data. For detecting a burst in an entity’s time-series data, detecting bursts based on deviation from the mean across the training period was used in Balog Krisztian et al. (2013), while an approach based on moving-average burst detection was introduced in Wang, Zhang, et al. (2014).

Document-entity features are often based on measuring the similarity between an entity’s existing knowledge-base profile and the textual content of the document in question. The earliest attempts at document-entity similarity in Balog Krisztian et al. (2013) measures similarity using cosine similarity, jaccard similarity and KL-divergence between the Wikipedia page of an entity and the textual content of a document. Using the same similarity measures on textual content of existing citations on an entity’s Wikipedia page was also introduced in Wang, Song, et al. (2013). Later in the 2014 TREC KBA track entity profiles from Wikipedia were not available for all entities and an approach to building an entity profile automatically based on concatenating documents in the training data known to be citation-worthy was introduced in Wu et al. (2014).

A common way of building new supervised learning systems for CCR is to use a basic set of features which usually consist of the features introduced in Balog Krisztian et al. (2013) and then adding novel features on top of this feature set, two approaches that exemplify this are Wang, Song, et al. (2013) and Reinanda et al. (2016). This basic feature set together with some various other features were used in a “feature-aware comparison” in Gebremeskel et al. (2014), where they analyze features commonly used in CCR and conclude using a model trained on a small, carefully selected set of features can outperform one that uses a larger, less carefully selected feature set.

3.2 Knowledge base population

Cumulative citation recommendation is generally considered part of the broader research area of knowledge base acceleration. Because of this, work on cumulative citation recommendation is naturally related to work on knowledge base population (KBP) where the aim is to build technologies “that use unstructured text to populate knowledge bases about named entities” (Mitchell, 2013). One specific form of KBP where the relationship to CCR is especially apparent is in stream slot filling (SSF) which was ran as a task in the TREC KBA track alongside the CCR task in the years 2013 and 2014. The goal of SSF is to fill in predefined “slots” containing information about an entity, these slots can be things such as where a person was born, what year they were born in or where they currently live.

In the 2013 TREC KBA track the idea was presented that slot-filling for entities in SSF may be a good way of solving the CCR task as well, by the logic that a document that fills a slot for an entity will also be citation-worthy for that entity. In the 2013 TREC KBA track all SSF systems are also expected to solve the CCR task by providing each document that fills a slot with a confidence score indicating how likely it is to be vital. This has led to some interesting systems that perform both the SSF and CCR task at the same time, notable works include Nguyen et al. (2013) which assigns a confidence score based on number of variations in slot values found. As for the value in combining SSF and CCR the results have been mixed: In the 2013 TREC KBA track overview paper (John R. Frank et al., 2013) the organizers claim that they anticipate that eventually SSF systems will have the highest scores for the CCR task, however in the same overview paper they also show that SSF systems overall have significantly worse performance in the CCR task compared to dedicated CCR systems, indicating that bridging the gap between SSF and CCR isn't trivial.

TREC Cumulative Citation Recommendation

In this chapter we will describe the context and definitions used when performing CCR in the context of the TREC KBA track. In Section 4.1 we introduce the TREC KBA track. In Section 4.2 we introduce how the task of CCR is described in the TREC KBA track and the associated definitions. In Section 4.3 we introduce the StreamCorpus document collection and associated truth data used in the TREC KBA track. Finally in Section 4.4 we review the evaluation metrics used in the TREC KBA track.

4.1 The TREC KBA track

The Text Retrieval Conference (TREC) ran a Knowledge Base Acceleration (KBA) track between between 2012 and 2014 with the goal of examining issues related to creating systems that can participate in the process of assimilating information into knowledge bases. On their website their official goal is stated as follows: *Given a rich dossier on a subject, filter a stream of documents to accelerate users filling in knowledge gaps*¹.

The TREC KBA track has in all the three years it was active included a task related to CCR, in 2012 and 2013 this task was simply called the “cumulative citation recommendation task” while in 2014 the task name changed to the “vital filtering task”. Other than the name change the task was still to create CCR systems, for this reason we will use the terms “cumulative citation recommendation” and “vital filtering” interchangeably in this thesis. To allow for creation and assist in the equal evaluation of cumulative citation recommendation systems the KBA track organizers have created a document collection called StreamCorpus, and collected human generated truth data for documents in this collection.

¹<http://trec-kba.org/>

4.2 Task description and definitions

In order to properly approach the TREC CCR task one first needs to understand the context of the task and the specific terminology used to define different aspects of the task. The goal of the TREC KBA CCR task is to create systems which can filter documents that contain information about an entity in a knowledge base that would require that entity's profile in the knowledge base to be updated. Below we give a more in-depth description of the different aspects of the task.

4.2.1 Definition of an entity

An entity in the broader sense of KBA can be anything that can have a profile in a knowledge base. This can be as broad as any human concept such as “teapot” or “skydiving”. Obviously targeting all of these concepts would be difficult, and as such the TREC KBA track only cares about a carefully selected set of “target entities”. For the 2014 TREC KBA track a set of 109 target entities were chosen, however for the CCR task only 74 of these are relevant. Since we only care about these target entities in the task, we can redefine entity to only be the three entity types that are considered in the task: People, organizations and buildings. Table 4.1 gives an overview of the different entity types and their representation in the entity set. It is also important to note the selection of target entities for the task was not random: Entities “in the region between Seattle, Washington, and Vancouver, British Columbia” were selected as target entities for the task (John R Frank et al., 2014).

Entity type	Count
Person	59
Building	5
Organization	10

Table 4.1: Target entity types for the 2014 CCR task and the number of entities in each category

4.2.2 Judging citation relevancy

In order to properly filter documents that would require an update to an entity's profile there needs to be a formal definition of how we determine if a document would require such a change. In order to standardize this definition for the TREC KBA track the organizers have defined four different relevancy levels which determine if a document contains information that would require a profile update. These relevancy levels are formalized in the “Assessor's guidelines”² which were created for human assessors to create truth data for the CCR task and are defined as follows:

²<http://trec-kba.org/data/index.shtml>

Vital documents would require a change to an already up-to-date knowledge base profile. The vital classification imply that the entity took an action, ie. participated actively in an event. Another criteria for a document being vital is that the information it contains is “timely”, ie. the event it describes did not happen a long time ago. While information is required to be timely, there is no novelty requirement for a document to be vital: If 100 documents describe the same vital event at different points during the day the event took place, then all of them are vital, not just the first document describing the event.

Useful documents contain information that would be relevant to include in an entity’s knowledge base profile, however the information is either not timely (ie. it is biographical) or the entity did not actively participate in the event. An example of this is that a document saying that a person was recently elected to a position is vital, while a document simply stating that the entity has an elected position is useful. One way to describe the boundary between vital and useful documents is that vital documents are relevant both when we have an existing, up-to-date profile for the entity, while a useful document is only relevant if we are building a knowledge base profile from scratch, and need to include old biographical information in the new profile.

Unknown documents contain biographical information about an entity, but the text make it difficult to disambiguate the entity in question. For example if a document contains biographical information about a “John Smith” but it is not clear from the text which John Smith is described it is classified as unknown.

Non-referent documents are similar to unknown ones except that it is clear from the context that it is a different entity than the one we are targeting that is described. For example if a target entity is called “Jogn Smith” but we determined that the document describes an event relating to *another* John Smith then the document is non-referent.

4.2.3 The no future information rule

To ensure the proper and equal evaluation of the participating systems in the TREC KBA track there are some additional technical requirements that must followed by participants of the TREC KBA track: Since each system is determining relevancy of documents in the past, it is possible for systems to “cheat” by gathering facts about the documents or entities which would not have been known at the time of evaluation.

For this reason the TREC KBA track has a “No future information” rule which require systems to judge relevancy for each document using only information that was available when the document in question was created. For example a system may not use an entity’s Wikipedia page from 2014 to determine if a document from 2012 is relevant for that entity. Additionally a system may not go “back in time” to update judgments of documents using information it has learned judging later documents. While systems may not access future information, they can however iterate over the document collection by date and update their knowledge as time passes, a system may for example process documents in yearly batches and retrieve updated version of Wikipedia pages each time it processes a new year.

4.3 Document collection and truth data

In order to evaluate the performance of CCR systems a large collection of documents is required to perform the task on. Human generated judgments on the documents in this collection is also needed in order to provide a “ground truth” with which we can evaluate a system’s own judgments against.

4.3.1 StreamCorpus

StreamCorpus is a series of document collections created specifically for use in the TREC KBA track. For each of the years the TREC KBA track has been active, a new iteration of this collection has been created. Where each iteration adds more documents to the collection. For the 2014 TREC KBA track the 2014 StreamCorpus collection is used which consists of over 1.2 billion documents scraped from different parts of the Internet.

Documents contained in StreamCorpus are scraped from a variety of sources on the Internet which includes, but is not limited to, forum posts, blog posts, news articles, and scientific articles. Table 4.2 lists the relevant fields contained in each StreamCorpus document for this thesis. Arguably the most important field is the “cleaned_visible” which is usually used as the source of the textual content of documents in most systems.

Document field	Description
doc_id	Identifier of the document, which may change over time, and of which this stream item is a snapshot. This is always an MD5 hash of abs_url.
raw	The original download as an unprocessed byte array.
clean_html	HTML-formatted version of raw. This has correct UTF-8 encoding and no broken tags.
clean_visible	Copy of clean_html with all HTML tags replaced with whitespace.
tagging	Set of auto-generated taggings, such as a one-word-per-line (OWLP) tokenization and sentence chunking with part-of-speech, lemmatization, and NER classification.
sentences	A dictionary mapping tagger IDs to ordered lists of Sentence objects produced by that tagger.

Table 4.2: Subset of fields in a StreamCorpus document which we use in this thesis. Descriptions adapted from streamcorpus.org

In the 2014 TREC KBA track a problem that became apparent was that StreamCorpus had grown so large that it was difficult for participants to process it without significant computational resources available (As illustrated in Figure 4.3). Because of this a filtered collection was created for the 2014 TREC KBA track. This filtered collection is simply a filtered version of the original document collection where only documents that mention one of the target entities of the 2014 TREC KBA track have been kept in the collection. This filtering significantly reduces the size of the document collection and allowed for easier processing of StreamCorpus for participants of the 2014 TREC KBA track. This size difference is illustrated in Figure 4.3, where we can see that the 2014 (filtered) collection contains significantly less documents than the unfiltered 2014 collection.

Year	Document count	Timespan	Start month	End month
2012	>400 Million	7 months	Oct. 2011	Apr. 2012
2013	>1 Billion	17 months	Oct. 2011	Feb. 2013
2014	1.2 Billion	19 months	Oct. 2011	Apr. 2013
2014 (filtered)	20 Million	19 months	Oct. 2011	Apr. 2013

Table 4.3: Evolution of the TREC KBA StreamCorpus between the 2012 and 2014 KBA track

4.3.2 Truth data

As mentioned earlier the goal of the CCR task is to filter documents based on their importance in building a knowledge-base profile for certain entities, the truth data for the CCR task is therefore comprised of human generated judgments on the importance of certain documents in StreamCorpus to each target entity’s hypothetical profile. The process for generating this truth data is comprised of two steps: First mention extraction was performed on each document in StreamCorpus to identify which of the target entities are mentioned in each document, effectively producing a list of document-entity pairs. Then each document-entity pair was classified into one of the four relevancy ratings presented in Section 4.2.2 by human annotators. The result of this is a set of document-entity pairs where each has been annotated by at least one human annotator. This means that each document-entity pairs may have several different ratings in truth data, which is the result of two or more annotators disagreeing on the rating of a document-entity pair.

In John R Frank et al. (2014) they review the rate of annotator agreements on judgments in the truth data for the 2014 TREC KBA track, and find that for the “vital” rating there is a 67.70% agreement on which document-entity pairs are vital in the truth data.

For aiding in the creation of CCR systems the document-entity pairs in the truth data has been split into a training- and testing-set. The decision of which document-entity pairs goes into which set was decided by defining a training-time for each entity: The training-time for an entity is a date such that all truth data before that date can be used freely as training data for CCR systems, all truth data after this date may only be used for evaluating these systems. The training-time was defined per entity such that each entity should have at least 20% of its “vital” annotated document-entity pairs in the training set. The result of this is that each entity has their own training-time, and that these dates may vary by almost a year for some entities.

4.3.3 Potential problems of working with StreamCorpus and truth data

Because StreamCorpus is a very large document collection with document collected from a variety of sources, we discovered certain quirks of the collection we feel need to be addressed in order to work with it properly. The first quirk is the amount of duplicate documents contained in the collection: In our initial exploration of the StreamCorpus collection we found that many of the documents it contains are duplicate or near-duplicates of documents earlier in the collection. Furthermore some duplicates are annotated differently in the training set, meaning two document can have exactly the same or close to equivalent textual content and yet be annotated differently, while we cannot confirm this, we believe this is because if an annotator sees the same document twice at very different times, the annotation guidelines say that the later one should not be vital due to the content not being timely any more. We believe handling this problem could be an important factor in training supervised approaches.

Another quirk can be found in the description of the `cleaned_visible` field which we will use as the basis for document textual content. Since the description says it only removes HTML tags leaving effectively everything else, this means the `cleaned_visible` field can contain significant amounts of text which is derived from sources such as hyperlinks and text in the header and footer of the original website. This problem is illustrated in Figure 4.1, as we can see there is a significant amount of text in the `cleaned_visible` field which is unrelated to the actual article content. Obviously having noise in the form of irrelevant text in the documents can cause problems for systems which use this textual content to make judgments.

Finally the number of available training examples per entity in the training data is highly imbalanced. To better illustrate this we have listed the number of training examples for each relevancy level per entity in Table A.1. As we can see from this table, when we aggregate the total number of training examples for each entity some have as few as 2 training examples in total for some entities while others have over 600. Furthermore there is a strong class imbalance for some entities such that one vitality rank can represent over 90% of the total training examples for that entity. Some relevancy levels are also more applicable to some entities than others: The entity “Rick Hansen” in Figure A.1 for example is much more prone to non-referent documents than most other entities. Obviously this class imbalance can affect the performance of models trained with this truth data, especially when building a single entity-independent model for all entities.

Attawapiskat kicks out federally appointed third-party manager

classifieds jobs cars obituaries celebrating shopping homes personals Weather Centre 4°C Unknown Nanaimo Detailed Forecast Nanaimo Home Search for Home & Site Index News Editorial Business Special Section Sports Driving Entertainment Columnists Nanaimo Daily News: 135 Years Live Green Obituaries Cars Contact Us Stocks Pages Carrier Application Back Issues Harbour City Star Latest from the Star Classified classifieds obituaries cars working (jobs) homes personals shopping Island Papers Spotlight Features Driving eFlyers Related Links

Canada's First Nations policies cause friction, human rights challenges: U.S. diplomatic cable Harper, First Nations plan summit on reserves It's a disgrace, not a surprise Aboriginals must

...

response to the plight of Attawapiskat's 2,100 residents, saying the community has received some \$90 million in government funding since the Conservatives took office in 2006.

achung@postmedia.com hroberts@postmedia.com © Postmedia News 2011 Ads by Google Inside the canada.com Network . Newspapers: National Post Victoria Times Colonist The Province (Vancouver) Vancouver Sun Edmonton Journal Calgary Herald Regina Leader-Post Saskatoon StarPhoenix Windsor Star Ottawa Citizen The Gazette (Montreal) DOSE Postmedia Community Publishing VING Newspapers: Alberni Valley Pennyworth Alberni Valley Times Campbell River Courier Islander Comox Valley Echo Cowichan Valley Citizen Harbour City Star Nanaimo Daily News Oceanside Star Tofino-Ucluelet Westerly Marketplace: working.com driving.ca remembering househunting shopping Swarmjam Sweet Deals About canada.com Site Map FAQs Privacy Terms Contact Us Copyright & Permission Rules © 2009 Postmedia Network Inc. All rights reserved. Unauthorized distribution, transmission or republication strictly prohibited.

Figure 4.1: Example of StreamCorpus document containing noisy text unrelated to the article (bold)

4.4 Evaluation metrics

The official evaluation metric for the CCR task is the maximum macro-averaged F_1 and SU measures which we described in Section 2.4. The macro-averaged metric works by

calculating the recall and precision for each target entity separately and then averaging across all entities. This can be computed as follows.

$$F_1 = \sum_{e \in E} \frac{F_1(e)}{|E|} \quad (4.1)$$

Where E is the set of all target entities of the task, $F_1(e)$ is the F_1 measure calculated from the only the truth data for entity e .

In the TREC KBA track CCR systems are expected to output a confidence score for each document-entity pair representing how confident the system is that the given document-entity pair is vital, the confidence value has to be reported in the range 1-1000 where 1 represents the lowest confidence and 1000 the highest.

The final F_1 measure for a run is calculated by sweeping a cutoff value τ across the possible confidence values from 1 to 1000. All judgments that have a confidence score higher than τ are considered “vital” while those with a lower value are considered “non-vital”. The macro-averaged F_1 measure is then calculated for each of these cutoff values and the cutoff value that achieves the highest score is chosen as the final macro-averaged F_1 for the run.

In addition to the maximum macro-averaged F_1 metric, the scaled utility (SU) metric that was described in Section 2.4.3 is also used to evaluate each run. This metric is also calculated by sweeping the cutoff value τ across the possible confidence values and taking the highest SU score as the final SU score of the run.

Implementation

In this chapter we will detail our implementation of a CCR system created with the purpose of answering the research questions we presented earlier. In Section 5.1 we present an high-level overview of our CCR system. In Section 5.2 we describe our approach to creating word-vectors which we will use for some features we implement later in the system. In Section 5.3 we give an in-depth description of the first stage of our system which deals with processing the document collection. In Section 5.4 we describe the second stage of our system which deals with ranking document-entity pairs by the vitality rankings defined for the TREC KBA track.

5.1 Architectural overview

The high-level architecture of our system is shown in Figure 5.1, as can be seen from this figure our system consists of two subsystems which run in sequence:

The “Document processing system” is the first stage in our system and is engineered to deal with the large size of document collection used for evaluating CCR systems in the context of the TREC KBA track. Its task is to detect which documents in the collection mention which target entities and produce document-entity pairs from this. For each document-entity pair this system also extracts the relevant features from the document and combine these into a single feature vector. The resulting output is all detected document-entity pairs and their respective feature vectors.

The “Vital filtering system” is the second stage of our system and deals with assigning a confidence score to the resulting document-entity pairs using the feature vectors that were produced in the first stage. This stage uses a training set of document-entity pairs to train an LTR based model which can then make predictions on the relative ranking of document-entity pairs. The final output of this stage is the predicted scores of all document-entity pairs produced by the previous stage.

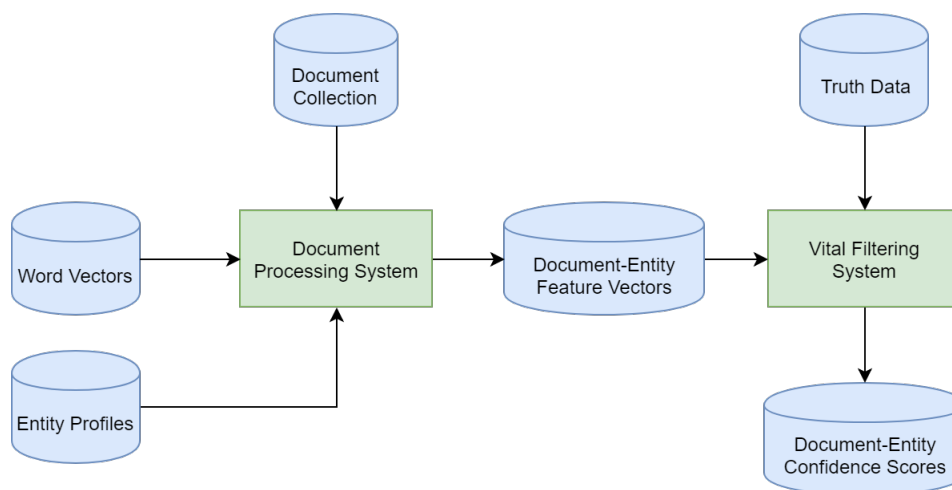


Figure 5.1: High-level architecture of our system

5.2 Building word vectors for temporally sensitive tasks

In tasks where the “no future information” rule is irrelevant the usual way to acquire word vectors for research is to download pre-built vectors from reputable sources. However since we cannot use future information in the CCR task all pre-built vectors we have found are unsuitable because they do not guarantee that they were not trained on documents that may be from the future in relation to some of the documents we need to classify. The reason this is problematic is that these vectors may have “learned” relationships between words that should not yet be known to our system. As an example word-vector models built after 2008 will learn there is a strong association between “Barack Obama” and the term “president”, obviously this will give this word-vector model an unfair advantage when used for ranking documents prior to 2008. From the perspective of the 2014 TREC KBA track task we cannot use any model trained on documents containing information that appeared after 2011-11-05 since this is the timestamp of the first documents in our testing collection.

For this reason we train our own word vectors using only documents from before 2011-01-17. This guarantees that the vectors we use are not trained on relationships that should not exist when ranking any of the documents in our testing collection. Our training corpus consists of all the English Wikipedia articles dumped in January 2011 and all documents in The New York Times Annotated Corpus. For English Wikipedia articles we used Wikiextractor¹ to extract the article text from the article dump we used. For the NYT corpus we extracted the document body from each annotated article from 1996 to 2007. Table 5.1 summarizes relevant statistics of our training collections, the resulting training collection contains 2.3 billion tokens. For comparison the collection used to train word vectors in (Pennington et al., 2014) consist of between 1 and 42 billion tokens.

¹<https://github.com/attardi/wikiextractor>

There are many different approaches to training word vectors as described in Section 2.3.2. Initially we considered embeddings based on Word2Vec, GloVe and Doc2Vec. Doc2Vec turned out to be unusable because it cannot train paragraph vectors in an online manner, this means that we would have to train paragraph vectors for all testing documents at the same time which breaks the “no future information” rule. The choice between GloVe and Word2Vec was harder because the two approaches have achieved very similar results on NLP related tasks, in the end we went with GloVe because it has been shown to give slightly better results compared to Word2Vec (Pennington et al., 2014), however the two methods should produce similar results.

When training our word-embedding model we use publicly available toolsets for GloVe². We train 300 dimensional models, and use the default parameters except for the iteration count which we increase from 25 to 100 in order to match the iteration count used for high-dimensional vectors in Pennington et al. (2014).

Document source	Num. tokens	Newest document timestamp
NYT Corpus	1B	June 19, 2007
English Wikipedia	1.3B	January 17, 2011
Combined	2.3B	January 17, 2011

Table 5.1: Statistics of different collections which we combine and use to train our word-embeddings.

5.3 The document processing system

In Figure 5.2 the different steps of the document processing system are shown: Effectively this system is a pipeline that iterates over each document in the output of the previous stages. In this section we will describe how each of the steps in this system work.

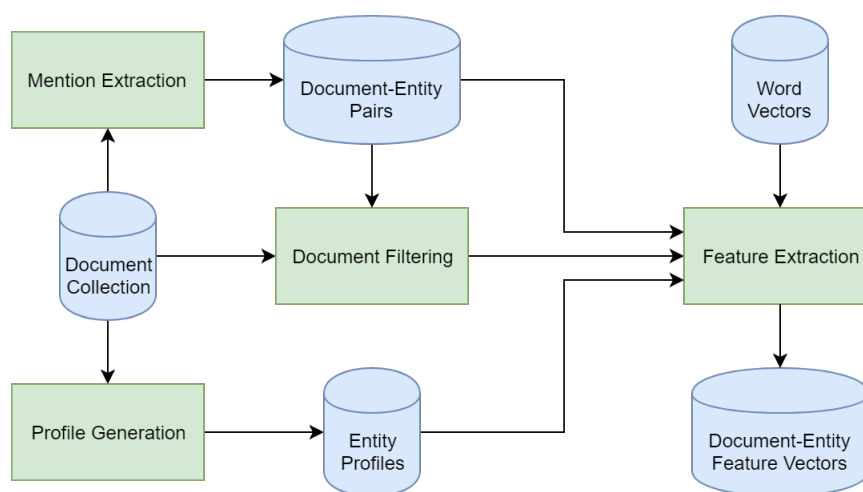


Figure 5.2: Architecture of the document processing system

²<https://github.com/stanfordnlp/GloVe>

5.3.1 Mention extraction

The first problem we face when solving the CCR task is identifying which document potentially mention one of the target entities, since we are ranking document-entity pairs and not individual documents we need a way for our system to detect candidate document-entity pairs that which represent documents that *might* be relevant to an entity. Since the document collection contains over 20 million documents it is not feasible to apply any computationally expensive algorithm to determine which document mention which entities. Our strategy for mention extraction is therefore relatively simple: For each entity we collect a set of “surface-forms” of the entity, which is simply the different names the entity could be referred to by. We then look for these in the document text to identify if a target entity was mentioned in document.

Common surface forms for people would be their first-name, last-name and various combinations of these, nicknames and various abbreviations of their full-name can also be considered surface-forms. For each entity we collect their surface-forms in two ways. First we include the “canonical name” of each entity which is defined in the truth data of the 2014 TREC KBA track. The canonical name of an entity is the full name of the entity, for example “John Smith” or “Norwegian University of Science and Technology”. Additionally some entities have entries in existing Knowledge Bases such as DBPedia³ where different naming variants are defined. For entities that have an entry in DBPedia we collect additional surface-forms, ignoring those that consist only of the entity’s first-name or last-name as these are too generic.

Using the aforementioned surface-forms we detect occurrences of these in the text by using a method similar to the one used in Balog Krisztian et al. (2013) where they do a simple substring match for surface-forms in the document text. Our approach works the same way except we introduce an additional requirement: If we match a surface-form it must also not be preceded or followed by a word character. This additional requirement was introduced because some entity surface-forms could be matched as part of partial, unrelated sentences. For example the canonical name of the entity “Ted Prior” would match with the string “Created Prior” if we did only a simple string match.

By iterating over the whole document collection and applying the method above to the textual content of each document, we produce a collection of candidate document-entity pairs. Each pair in this collection denote a single document mentioning a single target entity. A single document may therefore be part of multiple document-entity pairs where each pair denotes the document mentioning a different entity. These document-entity pairs play a vital role in for the rest of our system: Since in CCR we want to differentiate vital vs non-vital documents for different entities, a document may be vital for one entity while the same document may be non-vital to another, even if both entities are mentioned in it. This means that in our system we will be ranking these document-entity pairs as either vital or non-vital, and not individual documents.

³<http://wiki.dbpedia.org/>

5.3.2 Collection filtering

As mentioned previously we will be ranking document-entity pairs and not individual documents, because of this any document that does not participate in a document-entity pairs is essentially redundant and will not be used later in our system. Because of the large size of the StreamCorpus document collection it is beneficial to filter out any document which does not mention any of the target entities. To do this we simply produce a new document collection by iterating over the original collection and ignore any document that is not featured in any of the document-entity pairs we found in the “mention extraction” step.

5.3.3 Training profile generation

One of the unique challenges in the 2014 TREC KBA track is that a significant number of the target entities do not have training profiles in the form of Wikipedia pages which can be used as entity profiles in the CCR task. Existing approaches to the CCR task rely on external entity profiles for computing the similarity between an entity and a candidate document, something that can be used as a feature in CCR systems. Since we will later introduce features that require entity profiles we need a method of generating these entity profiles.

A solution to the lack of external profiles in the 2014 TREC KBA track was presented in Wu et al. (2014) which is based on combining the textual content of all vital training document for an entity into a single document which serves as a entity profile. In our profile generation algorithm we will use a similar approach with some modification which we believe will improve the quality of the resulting profiles, specifically we will introduce some additional document filtering in order to generate better entity profiles based on the problems with StreamCorpus we discussed in Section 4.3.3.

From the definition of vital we presented in Section 4.2.2 we know that for a document to be vital the entity needs to actively participate in the event described in the document. We therefore present a method of removing all sentences in a document that does not directly mention a given target entity. The idea behind this method of filtering a document is that the only textual content that we want to keep in an entity profile from a given document is the content which talks directly about the entity, and it is therefore these sentences we want to build the entity profile on.

The approach to sentence filtering uses the method of mention extraction presented Section 5.3.1: First we perform mention extraction on each document and filter on sentences that mention one of the entity’s surface-forms. When extracting mentions for the purpose of filtering sentences we make a small modification on the approach in Section 5.3.1: We noticed that when people are mentioned in documents there is almost always a mention of their full name ie. “Barack Obama” while following mentions use only the first-name or surname ie. “Obama” or “Barack”. Since we already know the full name of the entity is mentioned at least once for all documents remaining after the collection filtering from Section 5.3.2, we assume that any mention of a persons first-name or surname in the remaining documents also refers to that entity. Doing this allows us to extract more sentences from each document, reducing the likelihood we miss important information in the documents relating the an entity.

Our approach to filtering the content of a document is detailed in Algorithm 1, here $mentions(t, e)$ returns true if one of the surface-forms of entity e was mentioned in the text t .

Algorithm 1: Filter sentences mentioning an entity from a document

Data: Document d
Entity e

```
1  $filtered \leftarrow ""$ 
2 foreach  $sentence$  in  $d.sentences$  do
3   | if  $mentions(sentence, e)$  then
4   |   |  $filtered \leftarrow "" + sentence$ 
5   | end
6 end
7 return  $filtered$ 
```

In Wu et al. (2014) they build profiles using only vital documents from the training data, and from how they describe their approach, they use the raw document content without any filtering. As we discussed in Section 4.3.3 this is problematic because documents in StreamCorpus often contain irrelevant data as we discussed in Section 4.3.3. We will build an entity profile for each entity by combining the textual content of both vital and useful training documents, in our solution we include useful documents in addition to vital due to the definition of useful in the 2014 TREC KBA track: Recall that the definition of useful we presented in Section 4.2.2 says useful documents contain biographical information about an entity which is useful when building an entity profile from scratch, since we are effectively building a new entity profile we therefore argue that it is relevant to include both useful and vital documents, as opposed to only using vital ones.

Another issue we have to deal with when building entity profiles is that there is no novelty requirement for the 2014 TREC KBA track. As we mention in Section 4.2.2 StreamCorpus contains duplicates of the same document, that due to there being no novelty requirement may all be annotated vital. If nothing is done to mitigate this we can end up with entity profiles that contain the same exact sentences many times.

In order to handle the issue of duplicates we require a method that can efficiently detect duplicates the document collection, and prevent these from being added to the profile. To do this we looked at different clustering algorithms that allow us to cluster duplicates effectively. While there are many possible choices for clustering algorithm, our choice is constrained by the “no future information” rule in the TREC CCR task. Clustering algorithms that need to scan the whole document collection before clustering are inappropriate because they will use future information to decide on the clustering topology. Common clustering algorithms such as DBSCAN and K-means are therefore not applicable because of this. The type of clustering algorithm we need is a clustering algorithm that works in an online fashion. After researching possible candidates in existing literature we decided that the classic Single Pass clustering algorithm described in Frakes et al. (1992) would be a good fit: It only has a single intuitive parameter in the form of the similarity threshold t and it handles clusters in an online fashion, which means using it does not break the “no future information” rule.

As for determining the similarity threshold parameter t we set it intuitively to 0.9, ie. there must be a 0.9 document similarity for documents to be clustered together. Our choice of 0.9 as opposed to 1 is based on the fact that if there is a slight change to a document, such as a spelling correction or equivalent we still want it to be clustered with other near-duplicates, for this reason we set the threshold slightly below 1 such that our clusters can contain “near-duplicates” as well.

Our implementation of a Single-Pass clustering algorithm is detailed in Algorithm 2.

Algorithm 2: Single-Pass clustering algorithm

Data: Set of documents D
 Similarity threshold t

```

1  $C \leftarrow \emptyset$ 
2 foreach document  $d$  in  $D$  do
3    $closest\_cluster = null$ 
4    $highest\_similarity = 0$ 
5   foreach cluster  $c$  in  $C$  do
6     if  $S(d, c) \geq t$  and  $S(d, c) \geq highest\_similarity$  then
7        $closest\_cluster = c$ 
8        $highest\_similarity = S(d, c)$ 
9     end
10  end
11  if  $closest\_cluster == null$  then
12     $c = \{d\}$ 
13     $C \leftarrow C \cup c$ 
14  end
15  else
16     $closest\_cluster \leftarrow closest\_cluster \cup d$ 
17  end
18 end
19 return  $C$ 

```

The function $S(d, c)$ calculates the similarity between a cluster c and a document d , which we defined as follows.

$$S(d, c) = \frac{\sum_{i=1}^{|C|} S_{doc}(d_i, d)}{|C|} \quad (5.1)$$

When calculating the similarity between documents we use the simple dice measure with binary term weights which can be defined as follows.

$$S_{doc}(d_1, d_2) = \frac{2C}{A + B} \quad (5.2)$$

Where C is the number of terms in the intersection of terms appearing in both d_1 and d_2 , A and B are the number of terms that appear in document d_1 and d_2 respectively.

We chose this similarity measure because we will primarily be comparing documents by the filtered content generated by Algorithm 1. Since this content is relatively short we represent documents in clusters by the set of unique terms that appear in them which makes the dice coefficient with binary term weights a computational efficient similarity measure.

5.3.4 Feature Extraction

In this section we will introduce the different features we will use in our system, our reasoning why we chose these features and how we extracted them.

The input of feature extraction is the document-entity pairs and associated filtered corpus produced by the Document Processing System and any additional data we need for specific features, which in our case is word-vectors and entity-profiles. The output is a set of numeric features for each document-entity pair. Our approach to feature extraction is detailed in Algorithm 3, here `extract_features(d, e)` is a function which extracts function performs the actual extraction of features for a document-entity pair (d, e).

Algorithm 3: Feature extraction

Data: Document collection C

Set of document-entity pairs S

Result: Set of document-entity feature vectors

```

1 foreach document  $d$  in  $C$  do
2    $M \leftarrow \text{get\_mentions}(S, d)$ 
3   foreach entity  $e$  in  $M$  do
4      $F \leftarrow \text{extract\_features}(d, e)$ 
5     output  $(d, e) \rightarrow F$ 
6   end
7 end

```

Basic features

In order to have a basic set of features to use as a baseline we reproduce the features introduced in Balog Krisztian et al. (2013). This feature set includes a variety of features that are both relatively simple to extract and have been shown to perform reasonably well during the 2012 TREC KBA CCR task, and as such are suitable as a baseline that we can compare a larger feature set to.

A problem with reproducing this feature set faithfully is that many of the features in this feature set require entities to have Wikipedia profiles, which is problematic considering that many entities do not have Wikipedia profiles in the 2014 TREC KBA track. To deal with this the related entity features and the temporal features based on Wikipedia pageview data have been removed in our implementation because we have no substitute for the Wikipedia data these features are based on. The method of generating profiles using training documents that was described in Section 5.3.3 is used to generate profiles for each entity based on training data, this allows us to keep the similarity features although the system even though we do not have Wikipedia profiles for all entities.

The remaining features are extracted in the same way as in Balog Krisztian et al. (2013), the full feature set with descriptions is shown in Figure 5.2. For a more detailed description of the basic feature set, see Balog Krisztian et al. (2013).

Feature name	Description
$Len(d)$	Length of document body
$Src(d)$	Source of document
$Eng(d)$	Document language is detected as English
$FPos(d, e)$	Position of first mention of entity in document body
$LPos(d, e)$	Position of last mention of entity in document body
$FPos_n(d, e)$	$FPos(d, e)$ normalized
$LPos_n(d, e)$	$LPos(d, e)$ normalized
$Spread(d, e)$	$LPos(d, e) - FPos(d, e)$
$Spread(d, e)_n$	$Spread(d, e)$ normalized
$SV(e)$	Average hourly stream volume over training period
$SV(e, h)$	Average stream volume over the last h hours
$\Delta SV(e, h)$	Change in stream volume over the last h hours
$SB(e, h)$	Burst in stream in the last h hours
$ProfSim_{cos}(d, e)$	Cosine similarity between document d and e 's profile page
$ProfSim_{jac}(d, e)$	Jaccard similarity between document d and e 's profile page
$ProfSim_{kl}(d, e)$	KL-divergence between document d and e 's profile page

Table 5.2: Basic features

Semantic similarity features

Features based on measuring the similarity between a document and an external entity profile has been extensively used in previous supervised approaches to the TREC CCR task. To our knowledge the similarity measurements that have been used are cosine similarity, Jaccard similarity and KL-Divergence on the classic vector-space model representation of the full document body, which are all included in the basic feature that was described earlier.

Another type of similarity measure we have not found any attempts at using are semantic similarity measures that measure the semantic similarity between the entity profile and a document. A relatively new approach to semantic similarity measurements are measures based on using word-vectors to measure the distance between two documents. Initially our approach to calculating the semantic similarity between two documents was based on using Word Mover’s Distance (WMD) from Kusner et al. (2015), however this measure turned out to be problematic as we found it did not handle calculating similarity between variable length documents very well and the computation time was generally too high for calculating the similarity for a very large corpus. Instead we use a simpler approach based the mean of word-vectors document representation we presented in Section 2.3.2, specifically we will use D_{avg} from equation 2.1 as a document-representation and calculate the similarity between an entity profile and a given document using the cosine distance.

For calculating the similarity we will use the entity profiles generated using the method described in Section 5.3.3 to generate entity profiles that we compare each document to. For the document we will use the sentence filtering approach in Algorithm 1 to remove unrelated textual content from the document. We also apply stop word removal and case-folding on both the entity profile and document. Both the entity profile and document are represented using equation 2.1 and we then use cosine similarity on these representations to calculate the similarity between the two texts.

When using this method we are utilizing the fact that semantically similar words will appear close to each other in the vector-space, thus if the document text and entity profile text contain many semantically similar words, they will appear closer to each other than if the document text contains words that are not semantically related to those in the entity profile.

Kurtosis & Trend features

Kurtosis is defined as the forth moment of a distribution and has been applied in temporal IR to detect sporadic and spikey events (Kanhabua et al., 2015). Given that earlier approaches to the CCR task have utilized various methods of detecting bursts in an entity’s time-series we want to see if kurtosis can be used in the same fashion to detect burts in an entity’s time-series data. Calculating the kurtosis for a series of temporal values y_1, \dots, y_N with mean \bar{y} can be done as follows.

$$\text{Kurtosis}(Y) = N \frac{\sum_{i=1}^N (y_i - \bar{y})^4}{(\sum_{i=1}^N (y_i - \bar{y})^2)^2} \quad (5.3)$$

We retrieve the time-series data for an entity using approach described in Balog Krisztian et al. (2013). We count the number of documents that mention the entity for each hour h and create a time-series of these hourly values. We then collect the time-series values N hours before the document timestamp and calculating the kurtosis of this subsample of time-series values. Instead of estimating a single value of N we use the approach of Balog Krisztian et al. (2013) where they include features for different choices of N . For our approach we considered the values $N = \{16, 24, 48, 96, 192\}$, effectively giving us 5 features that are different parameterizations of the kurtosis feature.

A second temporal feature we wanted to investigate was the trend over the last N hour in the time-series of the entity. Specifically we want to quantify whether there is a positive, or negative trend in the number of mentions over the last N hours, and how strong this trend is. To quantify this we fit a linear regression line $y = b_0 + b_1x$ to the values in the time-series over the last N hours. In this case the slope of the regression line can be seen as a estimate of the trend in the time-series data. The slope is the derivative of the regression line, ie. $y' = b_1$. The value we are interested in is b_1 which can tell us whether there is a positive or negative trend in the last N hours and how strong this trend is. To estimate b_1 from the sample of mentions in the last N hours we use the least squares estimate, which in Walpole et al. (2012) is given by the following equation.

$$b_1 = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^N (x_i - \bar{x})^2} \quad (5.4)$$

We use the same approach as with kurtosis for computing the value of b_1 . For a document-entity pair we compute multiple values of b_1 for $N = \{16, 24, 48, 96, 192\}$ back in time and use each as a separate feature in our system.

The kurtosis and trend features we implement are listed in Table 5.3, it should be noted that each of these feature represents all features for each choice of N , meaning there are more features than two as shown in the table.

Feature name	Description
$Kurtosis(d, t)$	Kurtosis of e 's time-series t hours in the past
$Trend(d, t)$	Slope of regression line fitted to e 's time-series t hours in the past

Table 5.3: Temporal features

Duplicate clustering features

As we mention in Section 4.3.1 when working with the StreamCorpus collection one thing we noticed is that it contains a lot of duplicate or near-duplicate documents. These duplicate documents often appear many days or even weeks after the first instance of the document appeared in the collection. This in itself is not a problem, however we also noticed that these duplicates are also featured in the training and testing sets where the same duplicate documents often appear with different labels at different points in time. We believe the reason for this is the requirements that documents must be timely: Since some duplicates appear many weeks after the first instance of the same document, the timeliness requirements leads the human annotators to classify them as non-vital because they saw a similar duplicate earlier in the document collection.

Obviously this will confuse any supervised learning system that does not handle this: It will see many documents with the same textual content but with different labels, which may cause problems for the algorithm when making inferences based on these examples.

At first we looked into removing duplicates from the document collection in the pre-processing stage, but the problem with this is that we have no concrete metric for determining if we should remove a duplicate or not: By the hypothesis that it is the timeliness that causes duplicates to go from vital to non-vital, we would have to know exactly how long it takes for a document that is a duplicate of an earlier document to go from being vital to non-vital. Instead of trying to estimate this ourselves we adapt the time range feature from Jiang et al. (2014) to leverage the ranking algorithm to solve this for us: By creating features which allow the prediction stage of our system to determine which documents are duplicates and how much time has passed since the first version of the document was detected in the collection, we can enable it to learn when to consider a duplicate non-vital.

Our approach to extracting features to do this is based on first applying Algorithm 2 to cluster duplicates in the stream. Once we have determined which cluster each document belongs to, we measure how “stale” a document is by calculating time difference in hours between the document timestamp and the timestamp of the first document in the cluster it belongs to. A document with no other duplicates will have a time difference of 0, and as the time between the first document in the cluster increases, the feature value increases as well, giving us a measure of how stale a document is.

Document-representation features

In Chapter 2 we presented various methods of representing the textual content of a document as features in supervised learning setting. The methods we presented were the vector-space model and mean of word-vectors representation. In order to evaluate how document-representation features can contribute to CCR systems, we will implement different variants of this feature type using the aforementioned methods of representing a document.

Before we represent the textual content of a document using either of the previously mentioned document-representations, we need to describe our approach to filtering the document. We will use an approach to filtering document textual content that is inspired by the “action-pattern features” presented in (Jiang et al., 2014). The action-pattern features they present use the open information extraction framework Reverb to extract relation triples from documents and extract those triples where a target entity is either the object or the subject of the sentence. While they use a more specialized way to representing these verbs as features in their system, we will use a more generalized approach and as such use triple extraction only to filter documents down to the verbs that signify what action the entity took in the document.

Our approach to extracting verbs is based on an open information extraction framework called Stanford OpenIE (Angeli et al., 2015). This framework has been shown to outperform Ollie, another open information extraction framework, which in turn has been shown to substantially outperform Reverb (Schmitz et al., 2012). The Stanford OpenIE framework extracts relation triples from sentences in a document. When extracting verbs for a given document-entity pair, we look for relation triples where the entity is either the subject or object of the sentence, then we construct a new document which only consists of the verbs that represent what the entity did in the document.

Algorithm 4 describes our approach to verb filtering: The `extract_triples(s)` represents the call to Stanford OpenIE to extract triples from sentence s , `mentions(t, e)` is the same mention extraction function that was used in Algorithm 1. Extracting the verbs in a relation triple can be done through the use of the POS tagging of the relation text which is provided as part of triple extraction using Stanford OpenIE. By looking at the POS tags of each term in the relation we can identify which are verbs, allowing us to filter out non-verb terms in the relation.

Algorithm 4: Filter out verbs from sentences where target entity participates

```

Data: Document  $d$ 
        Entity  $e$ 
1   $filtered \leftarrow ""$ 
2  foreach  $sentence$  in  $d.sentences$  do
3      if  $mentions(sentence, e)$  then
4           $triples \leftarrow extract\_triples(sentence)$ 
5          foreach  $triple$  in  $triples$  do
6              if  $mentions(triple.object, e)$  or  $mentions(triple.subject, e)$  then
7                  foreach  $verb$  in  $triple.relation$  do
8                       $filtered \leftarrow "" + verb$ 
9                  end
10             end
11         end
12     end
13 end

```

In Section 2.3.2 we presented two document representations that allow us to represent the filtered textual content as features: The classic vector-space model and the word-embedding based approach of taking the mean of word-vectors from Equation 2.1, we believe it would be interesting to apply both of these separately to see which will be best suited for the task of CCR: On one hand the vector-space model might perform well because of its relative simplicity, on the other hand the mean of word-vectors approach may perform better because of its ability to represent semantic relationships in document text. In order to show how these compare we implement both representations separately so that we later can compare their performance.

One potential problem when using a document-representation consisting only of verbs using the vector-space model is that we may encounter verbs we have not seen in the training set before. When using supervised learning our model can only be trained on verbs that appear in the training set, and it will therefore not be able to make any judgments about unseen verbs encoded with the vector-space model. Here the word-vector based approach may have an advantage because semantically similar words will appear close in the vector-space, something the model can utilize to deal with verbs that have not been seen in the training set.

As a way of mitigating the issue of unseen verbs for the vector-space model, we will present a third variant which augments the vector-space model with hypernyms from a lexical database. The idea for this approach is based on Mansuy et al. (2006) where they show that adding features using Wordnet hypernyms can improve accuracy in document classification tasks. To assist the vector-space model in dealing with unseen verbs we propose a similar method of augmenting the verbs in each document with related verbs by using the lexical database Wordnet (Miller, 1995). In this approach we augment the existing document-representation in by walking the hypernym tree of each verb seen in the document and adding all hypernyms to the document-representation.

To illustrate the motivation behind our approach consider the hypernym tree in Figure 5.3: If the verbs triumph and win are indicators of vitality in the training set, and prevail is not represented in the training set, then a document containing the verb prevail will not be classified as vital by a classifier using only the document verbs as features. However if we add the hypernyms of prevail then the document will also contain win, which the model uses as an indicator of the document being vital.

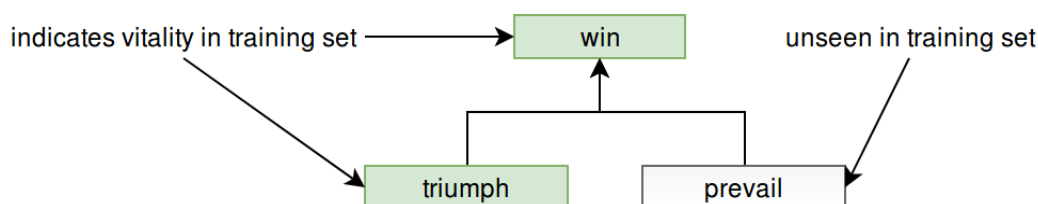


Figure 5.3: Partial hypernym tree of the verb win in Wordnet, verbs that indicate vitality in a hypothetical CCR classifier are labeled in green

Our approach to extracting the hypernyms is an extension of the verb filtering approach in Algorithm 4: After extracting a verb we look up the senses associated with the verb in Wordnet. For sense disambiguation two simple approaches that are commonly used with hypernym extraction are using all senses of a given verb or using only the most frequently used sense. We tried both of these approaches however we quickly found that the all-senses approach generated very large numbers of hypernyms for each verb and therefore settled on the most-frequent sense approach. Hypernym extraction is done by walking the hypernym tree of the sense until we reach the highest conceptual level. In the hypernym tree each node is itself a sense, we therefore add the identifier of that sense to the set of hypernyms of a given verb, effectively expanding the document textual content with the hypernyms of each verb.

Given the document filtering strategies and representations listed in this section, we can summarize the different document-representation feature variants we have implemented in Table 5.4.

Representation variant	Description
Verbs_VSM	Filter out verbs describing what the entity did in the document and represent verbs using VSM
Hypernyms_VSM	Filter out verbs describing what the entity did in the document represent verbs using VSM and supplement verb features with hypernyms
Verbs_AvgWV	Filter out sentences mentioning target entity and represent sentences using the mean of word-vectors representation

Table 5.4: Variants of document-representation features we implemented

Tense and clause features

By extracting relation triples we are also effectively determining if the entity is the subject or object of the the clause. This may be a useful feature because it tells us if the target entity is *acting* or *being acted upon* in the document.

When extracting the relation verb in Algorithm 4 we also get the POS tagging of the verbs from clauses where the entity is participating. One of the useful properties of POS taggings on verbs is that we can effectively tell if the event being described in the clause is happening in past or present/future tense.

The Stanford OpenIE framework we use to extract triples uses the Penn Treebank English POS tag set, in Table 5.5 we have listed the verb POS tags and which tense they indicate. The tense of the verbs can tell us if the event in a clause is described as happening in the past or in the present/future. Since the definition of a vital event requires it to be timely, this can help us tell if the document is describing something that has already happened, which makes the information biographical, and therefore useful instead of vital.

Tag	Description	Indicates past event?
VBD	Verb, past tense	Yes
VBN	Verb, past participle	Yes
VB	Verb, base form	No
VBG	Verb, gerund or present participle	No
VBP	Verb, non-3rd person singular present	No
VBZ	Verb, 3rd person singular present	No

Table 5.5: Verb tags from the Penn Treebank English POS tag set and the tense they indicate

The specific tense and clause features we implement are listed in Table 5.6, we also added a “skew” variant of each feature which may be better at indicating which variant is dominant in a given document.

Feature name	Description
$triplesPresentTense(d,e)$	Number of triples mentioning e that are in present/future tense
$triplesPastTense(d,e)$	Number of triples mentioning e that are in past tense
$triplesEntObject(d,e)$	Number of triples mentioning e where e is the object of the triple
$triplesEntSubject(d,e)$	Number of triples mentioning e where e is the subject of the triple
$tenseSkew(d,e)$	$triplesPresentTense(d,e) - triplesPastTense(d,e)$
$clauseSkew(d,e)$	$triplesEntSubject(d,e) - triplesEntObject(d,e)$

Table 5.6: Tense and clause features

5.4 The Vital Filtering System

As explained in Section 3.1 there are many different approaches to filtering vital documents. Many of these approaches fall into the ranking or classification approaches and can further be split into entity-dependent and entity-independent approaches. In our implementation we decided to focus on a ranking based approach because they have been shown to perform better than a multi-step classification approach when using the same feature set (Balog et al., 2012).

The architecture of the Vital Filtering System is shown in Figure 5.4. It is comprised of two main stages: Training and Prediction. In the training stage a ranking model is trained using a combination of truth data provided with the task and the feature vectors we extracted in the Document Processing System. The prediction stage uses the ranking model to rank the remaining feature vectors which we do not know the true label for. The resulting set of ranked Document-Entity pairs can be used to evaluate the system.

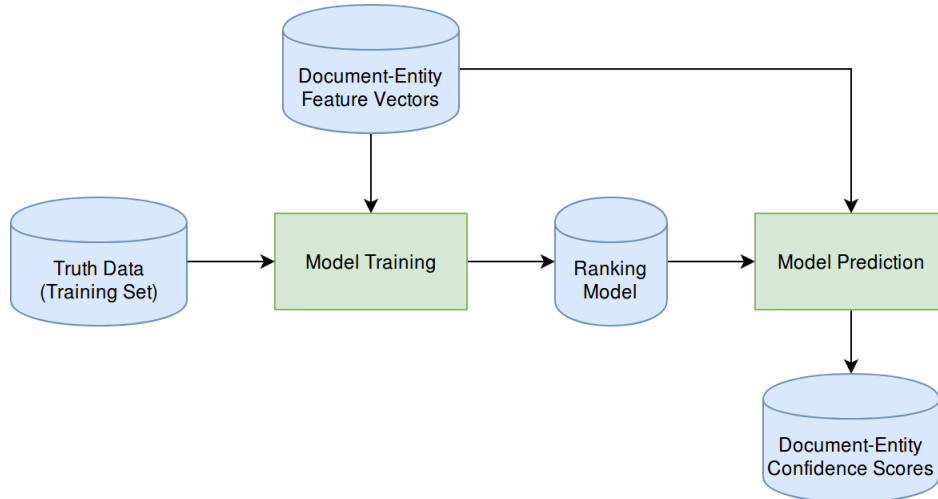


Figure 5.4: Architecture of the Vital Filtering System

5.4.1 Model training

The model training stage is responsible for using the training set provided in the 2014 TREC KBA track to train a ranking model which can rank unknown document-entity pairs relative to each other. Recall from Chapter 4 that there are four relevancy levels a document-entity pair can fall into: Vital, Useful, Unknown and Non-Referent. When using a ranking model in CCR, these levels are considered as relative document ranking in a IR ranking problem such that $\text{Vital} > \text{Useful} > \text{Unknown} > \text{Non-Referent}$. Under this assumption we can create a model to rank document-entity pairs relative to each other using any off-the-shelf LTR library given that we have feature vectors for each document-entity pair.

In existing research on CCR the de-facto standard library to train ranking models with is RankLib which was used both for the first LTR approach to CCR in Balog et al. (2012) and also the best performing solution to the 2014 TREC KBA CCR task in Jiang et al. (2014). Both Balog et al. (2012) and Jiang et al. (2014) use RankLib with Random Forests as the underlying machine-learning algorithm. While RankLib + Random Forest has achieved good results, we believe it would be interesting to look at a newer LTR library in our implementation as well, we therefore implement an alternative ranking algorithm which uses Gradient Boosted Trees using the XGBoost library (Chen et al., 2016).

When performing the actual model training in our implementation we have created a system where we can easily switch between ranking using either Gradient Boosted Trees with XGBoost and RankLib with Random Forest. Both of our implementation use the same underlying feature set which consist solely of numerical features and also the same training set. This means outside of the specific algorithms and their parameters there is no difference in how ranking is performed between the two models. When training either the Random Forests or the Gradient Boosting based implementation we use both with their default parameters which was done to prevent us from favoring either model unwittingly by tuning parameters on either models better than the other.

When performing training and prediction with LTR as discussed in Section 2.1.3 the underlying algorithm only compares ranks in query groups. For CCR there are no direct “queries” but since the point of ranking in groups is that the ranked documents in one query are not comparable to the ones in another query we see that the obvious analog for queries become the target entities themselves: Since the point of CCR is to recommend edits to specific entities then it also follows that recommendations for one entity are not comparable to those of another. For this reason we group the document-entity pairs in the training data by their respective entities. This way when we train our model the LTR algorithm only considers ranking between document-entity pairs that came from the same entity.

5.4.2 Prediction and Scoring

With the trained model produced in the model training stage the system can now predict the rank of unknown document-entity pairs in the document collection, this is done by applying the ranking model to batches of document-entity feature vectors grouped by their respective entity. The result are groups of documents ranked relative to each other such that a document which has a higher rank than another document from the same entity will be predicted to have higher likelihood of being vital to that entity.

As mentioned in Section 4.4 the output format of the TREC KBA track is such that all document-entity pairs must be ranked by a confidence score in the range 1 to 1000. The prediction scores created by the model however are not scaled to this range, the system therefore needs to transform the rank of each document-entity pair into this range. Rescoring document-entity pairs is simple if we know the rank given to each document-entity pair. We can apply linear interpolation to rerank each document-entity pair such that the highest scoring instance receives a score of 1000 while the lowest receives a score of 1. To do this we first determine the lowest and highest rank in the set I as follows.

$$\text{Min} = \min_{i \in I} S(i) \quad (5.5)$$

$$\text{Max} = \max_{i \in I} S(i) \quad (5.6)$$

Where $S(i)$ is the ranking score given by our learning-to-rank model for document-entity pair i . Each document-entity pair is then given a final score in the range of 1 to 1000 by linear interpolation as follows.

$$\text{Score}(i) = 1 + \frac{S(i) - \text{Min}}{\text{Max} - \text{Min}} \cdot 999 \quad (5.7)$$

5.4.3 Entity-dependant ranking

So far we have described training a single, global model for ranking vitality across all our entities. An alternative ranking technique would be to train one model for each entity and ranking document-entity pairs using the relevant entity's model. This has the benefit that the models trained do not have to compromise in order to predict accurately across all entities: Each model can focus on the unique properties of the specific entity and ignore properties of the other entities which may not be relevant. This method of ranking may also have drawbacks which are not apparent in the entity-independent approach, the most obvious being less training data for each model, the other being that we may not be able to make predictions for new entities without existing truth data for the new entity.

When training entity-dependent models using truth data from the 2014 TREC KBA track the system has to deal with the problem of some entities having much less training data than others. In the 2014 TREC KBA track the number of vital training examples can vary significantly between entities as was described in Section 4.3.3. Obviously training entity-dependent models for target entities with very few training examples will not be beneficial. In order to handle entities with too few training examples, we apply the method used in Wang, Zhang, et al. (2014) where they use a entity-independent global model when there are too few training examples for a given entity to train a entity-dependent model. The threshold for when we fall back to the entity-independent model was chosen such that entities with less than 50 training examples use the entity-independent model, while all entities with more than this use a entity-dependent model trained only on its own training examples.

Figure 5.5 shows the ranking method we implement for entity-dependent ranking: Initially training examples are grouped by the their entity, then a ranking model is trained for each entity using either the Random Forest or Gradient Boosted Trees algorithm. When ranking unknown document-entity pairs the respective entity's ranking model is used to make the prediction.

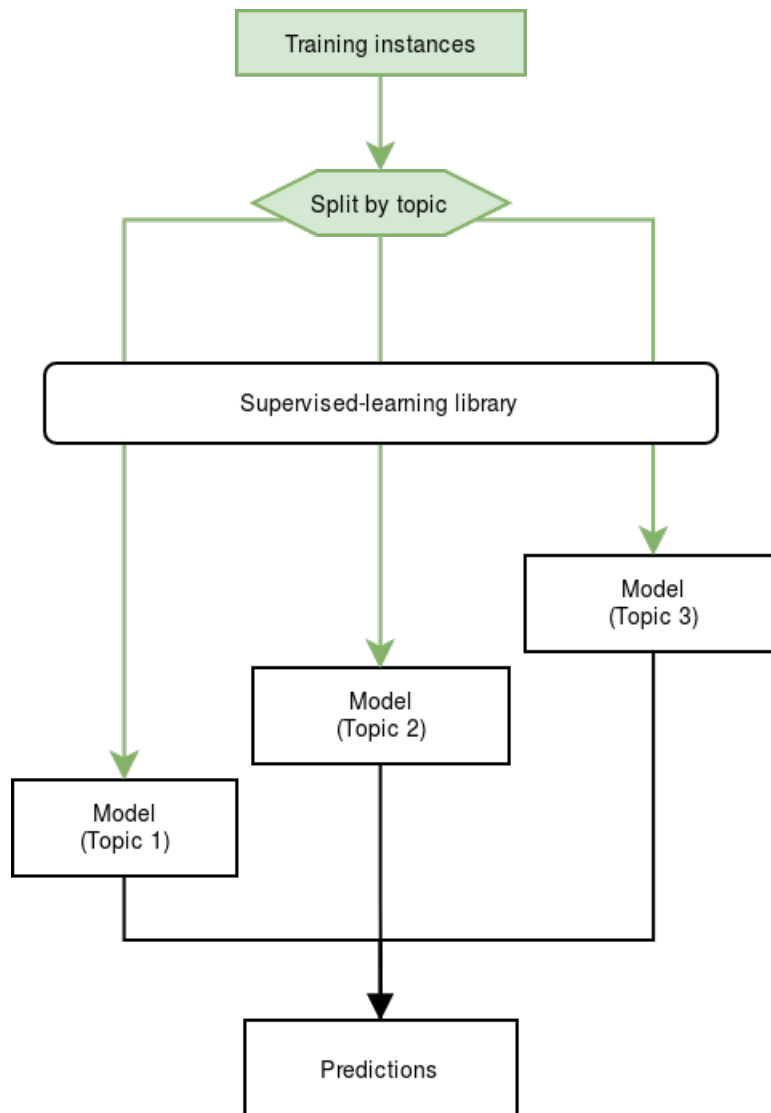


Figure 5.5: Architecture of the entity-dependant ranking approach

Experiments & Results

In this chapter we will describe how we evaluated the system we implemented in Chapter 5. In Section 6.1 we will present the document collection and truth data we used in our evaluation. We then present the evaluation metrics and how we extract these metrics in Section 6.2. In Section 6.3 we give an overview of the experiments we run and finally in Section 6.4 we will present the results of the aforementioned experiments with the chosen metrics.

6.1 Document collection and truth data

Our system was implemented to solve the CCR problem in the context of the TREC KBA track. Since the 2014 TREC KBA track is the most recent iteration of this track we will use the target entities, truth data and document collection that was used during the official 2014 TREC KBA track which we described in Chapter 4.

We retrieved the filtered 2014 StreamCorpus from the official TREC website¹. The truth data for the 2014 TREC KBA track which includes training set, testing set and listings of target entities was retrieved from the same website².

¹<http://s3.amazonaws.com/aws-publicdatasets/trec/kba/index.html>

²<http://trec-kba.org/data/index.shtml>

6.2 Evaluation methodology

When evaluating our approach it was important that we chose the appropriate metrics with which to evaluate our system. We evaluated some different choices of metrics based on earlier work on the TREC CCR task: The micro-averaged F1 and SU metrics were sometimes used together with the macro-averaged F1 and SU in earlier TREC KBA tracks, however micro-averaging is not frequently used for the 2014 TREC KBA track, where the macro-averaged F1 and SU metrics are the official metrics (John R Frank et al., 2014). Other than the F1 and SU metrics some other more novel evaluation methodologies have been presented in related research: In Gebremeskel et al., 2014 they use cross-validation to perform a comparison of individual features commonly used in approaches to the TREC KBA track, while in Dietz et al., 2013 they propose evaluating CCR systems in temporal batches to better quantify if a CCR system better or worse over time. While these metrics are interesting we were not able to find any other work having adopted these methods, this tells us it is likely more fitting to stick with more established metrics. In the end we decided that the most fitting metrics for our evaluation would be the macro-averaged F1 and SU metrics, ie. the official metrics of the 2014 TREC KBA track.

For generating the aforementioned metrics we use the official scoring tool for the 2014 TREC KBA track³, this tool allows us to easily retrieve both macro- and micro-averaged F1 and SU scores for runs we generate using StreamCorpus and associated truth data. The scoring tool was also used for generating the results for the 2014 TREC KBA track, allowing us to generate results in a way highly consistent with the official evaluation that took place in 2014. The scoring tool has different settings that must be properly configured for the evaluation to be correct. From our own preliminary work we have observed that changing even a single of these settings can widely change the results of a given system. We aim to run the tool with the same settings as was used for presenting the official results in John R Frank et al. (2014) where the official results of the 2014 TREC KBA track are presented. For posterity we have listed the specific command we used to run the scoring tool in Appendix B.

When evaluating the system performance we noticed that the official scoring tool for the TREC KBA track includes one entity from the entity list which is not a target entity of the CCR task. Specifically there appears to be a bug that causes it to consider the entity “_Nicholas.Tse” a target entity of the CCR task when this entity is only supposed to be a target entity for the SSF task. This is problematic because all CCR systems will have a F1 measure of 0 for this entity since they do not rank document-entity pairs for it, which has the effect that all runs have slightly lower F1 and SU score than they should for the CCR task. The bug seems to come from how the scorer tool handles entities in the testing set, by filtering the testing set such that it only contains annotations for the target entities of the CCR task we ensure the scorer does not use this entity when generating scores. Because of this all scores we report will be slightly higher than those reported elsewhere. Since we will not compare our results to anything we have not generated using the scoring tool ourselves, this should not be a problem.

³<https://github.com/trec-kba/kba-scorer>

6.3 Experimental overview

In order to answer our research questions we created experiments that show both the performance of the different features we implemented as well as the approaches to ranking. To summarize the ranking approaches we have implemented there is the entity-independent approach and the entity-dependent approach, for each of these there are also variations that use either Random Forests and Gradient Boosted Trees as the underlying machine-learning algorithm. This makes for a total of four distinct approaches we need to evaluate and compare.

As we mentioned in Section 5.3.4 we implemented a basic feature set which we used as a baseline feature set that we compare the performance of the rest of our features to. In order to evaluate how our system performed using the features presented in Balog Krisztian et al. (2013) which is a relatively old solution, we will first generate runs using only these features and then add the rest of the features to see how they affect the performance of the system. Table 6.1 shows how we partitioned features into groups which we evaluated separately to see how using different features affect the performance of the different approaches.

In order to show how the feature sets in Table 6.1 perform under each of the aforementioned approaches we split our experiments in two parts: First we evaluated the different features using both Random Forests and Gradient Boosted Trees using the entity-independent approach. We then performed the same set of experiments using the entity-dependent approach. This will both show us which feature set work with which approach, while also allowing us to compare the performance when using either the entity-dependent approach and the entity-independent approach.

Feature set name	Description
Basic	Only basic features adapted from Balog Krisztian et al. (2013)
Extended	All features except document-representation features
Verbs_VSM	Extended + Verbs_VSM from Table 5.4
Hypernyms_VSM	Extended + Hypernyms_VSM from Table 5.4
Verbs_AvgWV	Extended + Verbs_AvgWV from Table 5.4

Table 6.1: Variants of document-representation features we implemented

6.4 Results

In this section we present the results to the experiments we presented in Section 6.3. Our results are split into two parts: First in Section 6.4.1 we present the results for Gradient Boosted Trees and Random Forests with different feature sets under the entity-independent approach. In Section 6.4.2 we present the result of the same experiments when using an entity-dependent approach.

6.4.1 Performance of entity-independent ranking models

Table 6.2 together with Figure 6.1 shows the performance of different feature sets using the entity-independent approach to ranking. As we see from this we achieve a relatively good performance using only the basic feature set for both Random Forests and Gradient Boosted Trees. In terms of the F1 measure there is no significant difference in performance between the two algorithms, with both achieving equivalent or near-equivalent scores for all feature sets. When we look at the SU measure we see that there is slightly more variance in the results between the two algorithms. Specifically we see that the Random Forests algorithm has the best SU score for all feature sets except when using the “Verbs_AvgWV” feature set.

As for which feature set works best we see that using the “extended” feature sets improves performance in terms of the F1 measure. When adding different document-representations we see that we achieve our best results for both the F1 and SU measure when using the “Verbs_VSM” feature set, neither of the other two document-representations we implemented seem to give any notable increases in performance.

Feature set	Random Forest		Gradient Boosting	
	F1	SU	F1	SU
Basic	0.469	0.373	0.469	0.340
Extended	0.472	0.392	0.480	0.339
Verbs_VSM	0.486	0.398	0.486	0.343
Hypernyms_VSM	0.477	0.395	0.477	0.344
Verbs_AvgWV	0.453	0.350	0.454	0.355

Table 6.2: Performance of different feature sets when using an entity-independent approach

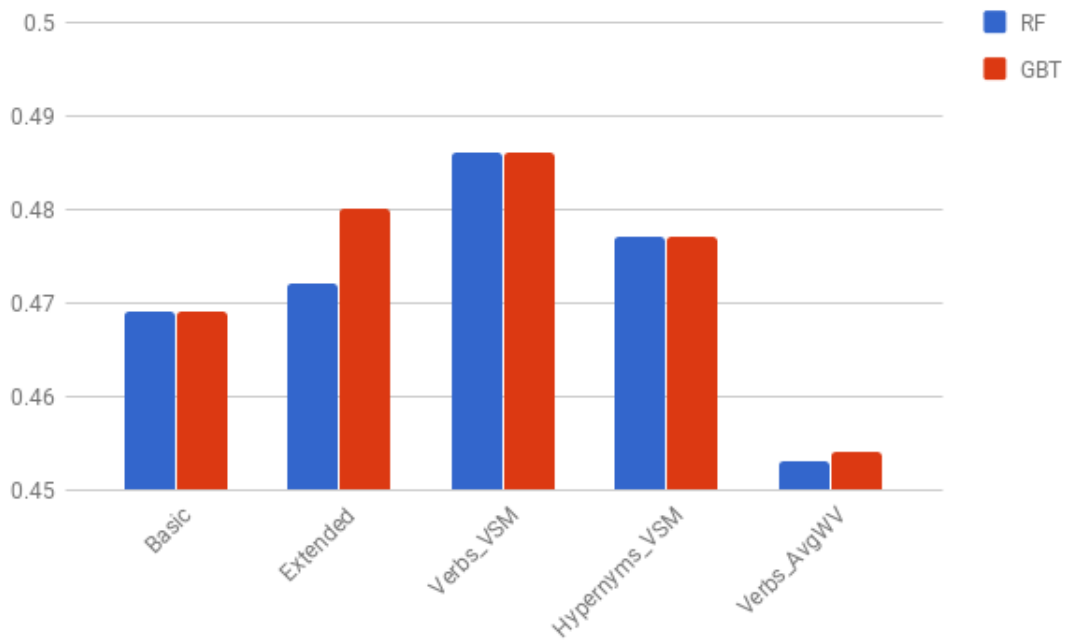


Figure 6.1: Graph showing the F1 measure of different feature sets and approaches when using an entity-independent approach

6.4.2 Performance of entity-dependent ranking models

Table 6.3 and Figure 6.2 shows the performance of different feature sets using the entity-dependent approach to ranking. For this approach we see that there is clear trend of favoring the Gradient Boosted Trees algorithm in terms of the F1 measure. For the SU measure the relationship is interestingly reversed, with the Random Forests algorithm generally performing better for this measure which mirrors what we saw in Table 6.2 for the entity-independent model.

As for which feature set performs the best we again see that the “Verbs_VSM” feature set achieves the best result in terms of the F1 measure. When using Gradient Boosted Trees and the entity-dependent approach with the “Verbs_VSM” feature set we achieve the best F1 score of this thesis of 0.5.

Feature set	Random Forest		Gradient Boosting	
	F1	SU	F1	SU
Basic	0.455	0.352	0.486	0.355
Extended	0.472	0.369	0.491	0.334
Verbs_VSM	0.468	0.366	0.500	0.344
Hypernyms_VSM	0.468	0.363	0.491	0.343
Verbs_AvgWV	0.458	0.355	0.454	0.337

Table 6.3: Performance of different feature sets and approaches when using an entity-dependent approach

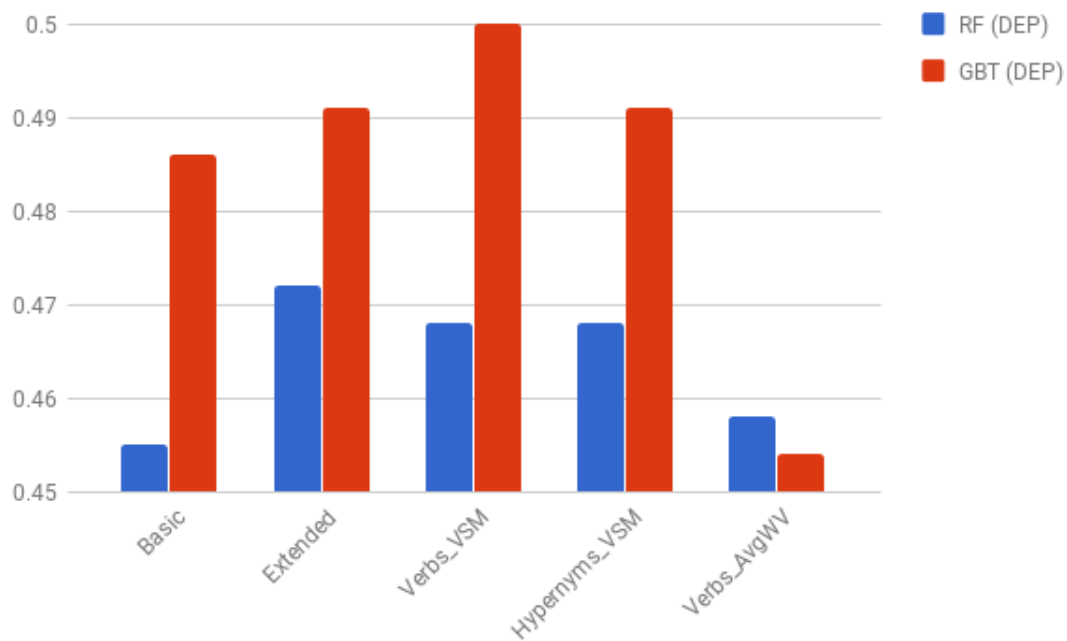


Figure 6.2: Graph showing the F1 measure of different feature sets and approaches when using an entity-dependent approach

From Figure 6.3 we can get a better overview of which methods perform best overall. We see that using Gradient Boosted Trees with the entity-dependent model has the best F1 measure for all feature sets except for the “Verbs_AvgWV” feature set. Using Random Forests with the entity-dependent approach is the weakest model overall, having consistently the lowest F1 scores except for when using the “Verbs_AvgWV” feature set where it is marginally better than the other approaches.

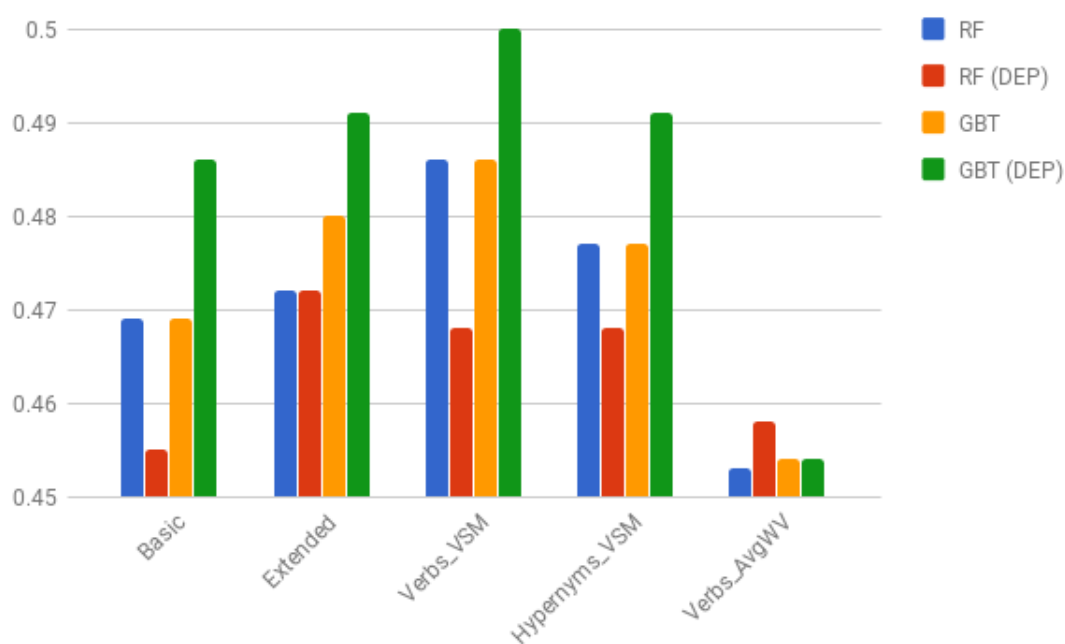


Figure 6.3: Graph showing the F1 measure of different feature sets for all approaches

6.4.3 Significance testing

Our results show that a model which uses an entity-dependent approach with Gradient Boosted Trees “Verbs_VSM” feature set performs the best of all our models. As we mentioned earlier our implementation using an entity-independent approach using Random Forests and only the “Basic” feature set is a close as possible replication of the approach presented in Balog et al. (2012) and therefore chosen as our baseline.

In order to test if our best approach significantly outperforms our baseline we perform a paired Student’s t-test on the scores of our best implementation and the baseline approach. This test was performed in accordance with the method presented in Section 2.4.4 with a standard significance criterion of $\alpha = 0.05$. The number of topics for the 2014 TREC KBA track CCR task is 74, we get a test statistic $t_0 = 1.9626$, the critical t value is $t(72 - 1, 0.05) = 1.9939$, since $t_0 < t(72 - 1, 0.05)$ we cannot say that our best implementation significantly outperforms our baseline implementation. The associated p-value is 0.05362.

6.4.4 Comparison with results from the 2014 TREC KBA track

We were given access to the raw runfiles of the official runs from the 2014 TREC KBA track which makes it possible for us to evaluate their runs in the same fashion as our own. In order to put our results in the context of other solutions to the 2014 TREC KBA track, we have in Table 6.4 listed the results in terms of the official metrics for both our best model and the best submitted model of each respective participating team in the 2014 TREC KBA track.

It should be noted that because of the bug we fixed with the scorer in Section 6.3, the results we generate for the participating teams of the 2014 TREC KBA track will all be somewhat higher than the ones published in other papers.

Team name	Best F1	Best SU
MSR KMG	0.573	0.518
This thesis	0.500	0.398
IRIT	0.453	0.356
uiucGLIS	0.452	0.379
BIT Purdue	0.450	0.345
KobeU	0.448	0.298
ECNU	0.410	0.345
UW	0.373	0.412
LSIS	0.287	0.393
WHU	0.272	0.372
SCU	0.244	0.335

Table 6.4: Best F1 and SU score of each participating team in the 2014 TREC KBA track and our best (bold)

As we can see from Table 6.4 our best run in terms of the F1 measure would have placed 2nd in the 2014 TREC KBA track. We can also see that the difference in terms of F1 score between our best and the best run of the track (MSR_KMG) is quite large with a difference of 0.073. It should however also be noted that the difference between our best run and the 3rd best run of the 2014 TREC KBA track (IRIT) is quite large with a difference of 0.047. The difference between our best and the 3rd best is so large that if we had used our baseline implementation based on Random Forests with an entity-independent approach and the “Basic” feature set we would still place 2nd.

Discussion & Conclusion

In this chapter we will first discuss the implications of our results in Section 7.1. Based on this discussion we present our answers to our research questions and a conclusion to this thesis in Section 7.3. Lastly we present ideas for future work in Section 7.4.

7.1 Discussion of results

In Chapter 6.4 we presented and described the performance of our system using the document collection and truth set of the 2014 TREC KBA track. In this section we will present the discussion of these results.

7.1.1 Choice of entity-dependent vs entity-independent approach

When we compare the results for the Random Forests algorithm vs Gradient Boosted Trees we see that for entity-independent models the performance of Gradient Boosted Trees is about the same as the performance of Random Forests based models as is shown in Figure 6.1. The gain in performance for Gradient Boosted Trees is first apparent when we switch from an entity-independent to entity-dependent approach as seen in Figure 6.2. Since the major difference between the entity-dependent and entity-independent approaches is that there are fewer training samples per model, this may indicate Gradient Boosted Trees is better at handling smaller training sets than the Random Forests algorithm.

Another aspect that may explain the problems of using Random Forests with the entity-dependent approach is the layout of training examples for each entity. As we show in Table A.1 some entities have very large class imbalances in their training data. If the Random Forests algorithm is badly suited to handle this large class imbalances then it will struggle with training entity-dependent models for entities where this problem is apparent.

If we disregard the results of the Random Forests algorithm and focus on Gradient Boosted Trees it is also interesting that this algorithm benefits so consistently from the entity-dependent approach. Given that the underlying algorithm and features are the same we would expect the same performance between these two models if there was no inherent benefit to entity-dependent ranking. However given that the entity-dependent model always outperforms the entity-independent model for Gradient Boosted Trees, this seems to indicate that there is an inherent benefit in entity-dependent ranking.

Considering that the primary trade-off when using an entity-dependent model is fewer training samples per entity for the benefit of needing only to target a single entity. This has the obvious benefit of the model being able to learn entity-specific features, something entity-independent models may struggle with considering they have to perform well for many entities. The question then becomes if this trade-off is worth it. Based on our results with Gradient Boosted Trees the answer seems to be yes, however the opposite is true when using the Random Forests algorithm, which makes it hard to make a specific recommendation. The result of the significance test show that our best entity-dependent model does not significantly outperform the baseline entity-independent model, which leads us to conclude that the choice between entity-dependent vs entity-independent is not going to significantly improve performance using a system similar to the one we implement. This result mirror results in related work, especially the results of Wang, Zhang, et al. (2014) which find that an entity-dependent model only slightly outperform an equivalent entity-dependent approach.

7.1.2 Choice of Random Forests vs Gradient Boosted Trees

Our results which compare the performance of the Random Forests and Gradient Boosted Trees algorithms is somewhat unclear given the the two algorithm have very difference performances when alternating between the entity-independent and entity-dependent approaches. Overall Gradient Boosted Trees achieve the best performance for most feature sets as shown in Figure 6.3, however this is only given that we choose to compare the two algorithms by the entity-dependent approach, if we use the entity-independent approach the results in terms of the F1 measure are much more equal.

When considering the choice of Random Forests vs Gradient Boosted Trees one should also consider the number of hyper-parameters that need to be tuned for each algorithm. Random Forests has always been a good choice when one cannot perform hyper-parameter estimation because it has relatively few hyper-parameters to tune. Gradient Boosted Trees, at least when implemented with XGBoost has many hyper-parameters which can play an important role in the performance of the model. One of the most important hyper-parameters in XGBoost is the gamma parameter which controls regularization and therefore reduces overfitting in the model, since we are using the default parameters we are using a default gamma of 0 which means we are not utilizing one of the strongest features of XGBoost.

Because of this we believe Gradient Boosted Trees implemented in XGBoost could achieve much better results than the popular RankLib + Random Forests approach if we properly estimated hyper-parameters. Unfortunately hyper-parameter estimation is difficult due to the fact that there is no validation set for the TREC KBA track which limits exploration of this idea.

7.1.3 Choice of features

From our comparison with other participants of the 2014 TREC KBA track we can see that the basic feature set even though being based on relatively simple features originally created for the 2012 TREC KBA track, can still perform admirably for 2014 TREC KBA track. This is exemplified by the fact that our system using only the basic feature set still places 2nd in the TREC KBA track. This results shows that this feature set is still a good basic feature set even in the 2014 TREC KBA track. The performance also indicates that the improved profile generation algorithm we implement is effective at generating entity profiles in the absence of existing Wikipedia profiles. Since our approach to reproducing the basic feature set does not depend on any Wikipedia data and still performs well it calls into question whether temporal and document similarity features that use Wikipedia data are actually needed in CCR. If this is the case then it would be beneficial if could remove Wikipedia based features without having to fear losing model performance because of it. Since the approach to generating entity profiles we implement is easy to generalize to entities that have no Wikipedia page it could be beneficial to simply drop the Wikipedia based features and use the profile generation approach we present.

Our second feature set is the extended feature set which adds more features to the basic feature set. As we saw in Figure 6.3 the extended feature set outperforms the basic feature set for all approaches to ranking and therefore seemingly contains useful features for CCR.

When we add different document-representations we see that that there is a clear winner in the simplest document-representation which is based on representing verbs from relation pairs that mention the entity with vector-space model and using simple binary weights. When we instead represent the same verbs with the mean of word-vector based representation we see that this variant has an overall negative impact on performance.

As for why the vector-space model representation on verbs outperforms the mean of word-vectors representation one reason could be that using vector-space model is simply a good fit for capturing the relevant information contained in the document: As we mention in Section 4.2.2 the criteria of vitality is that the entity must perform some action in the event described by the document. When representing the verbs from relation pairs with binary weights with the vector-space model it is trivial to determine if the entity has performed some action: We simply check if the feature value for a given verb is 0 or 1. When we want to extract the same information from the mean of word-vectors representation however the task is more complex: Since a verbs representation is distributed across multiple dimensions the task of determining if an entity has done something related to a specific verb becomes much harder.

The original idea behind using the mean of word-vectors was that the model would learn which dimensions in the distributed representation correspond to important verbs, and thus be better at handling verbs that are not seen in the training set. However given the model performance is much worse it is apparently not able to do this, which could be because we simply do not have enough training data to determine which dimensions are important. It would therefore be interesting to try the same approach with a much larger training set.

Furthermore the mean of word-vectors representation will obviously be affected by the quality of the underlying word-vectors we use to generate the representations for text. Since we train our own word-vectors the performance of word-vector based features will depend on the quality of this model: If the word-vector model we created is of low-quality, either because of a too small training corpus or mistakes on our part when we created them, then the performance of these features will suffer as well.

Our attempt at augmenting the vector-space model with hypernyms in order to better handle unseen verbs also does not appear to have a positive impact on model performance. As with the distributed representations our initial idea for this representation was that adding hypernyms would improve the model's ability to handle unseen verbs. As for why this does not work it may be that adding more verbs simply makes it harder for the ranking model to learn which specific verbs are important, which reduces the precision of the model. In this case the added hypernyms are then simply not utilized and instead act as noise to the ranking model.

As for which document-representation we can recommend based on our results, we can really only say that until more training data is available for CCR, using the vector-space model on verbs extracted from relation pairs is the most promising. When more training data is available it would be interesting to see if distributed representations are able to beat the simpler vector-space model.

7.2 Contribution

The main contribution of this thesis is the implementation of the CCR system we described in Chapter 5. The other contributions of this thesis come from how we use this system to answer the research questions we presented in Chapter 1. We will now revisit these questions and provide answers based on our results:

RQ1 How can we effectively filter out new, relevant information related to a set of entities given a stream of documents?

In this thesis we have explored several different approaches to creating LTR based ranking systems for CCR. Our results show that an LTR based system can achieve competitive results for the CCR track, which is exemplified by our system placing 2nd compared to the participants of the 2014 TREC KBA track. The fact that the system which placed 1st in the 2014 TREC KBA track is also an LTR-based system shows that LTR-based systems are currently some of the most effective CCR approaches which have been evaluated independently.

While LTR-based systems are currently achieving the best results for CCR we still question if their performance is good enough to be used in the real world. Currently no system we have seen have achieved an F1 measure above 0.6, indicating that there is still room for improvement. As we have not been able to achieve results close to this threshold with any of our LTR based approaches, we think researchers in the future should explore alternative solving the relevancy filtering problem of CCR.

RQ2 How does the Random Forest algorithm compare to Gradient Boosted Trees when used for ranking?

Our results show that Gradient Boosted Trees is a viable contender to the more popular Random Forests approach when using an LTR based approach. We are able to achieve our best results with Gradient Boosted Trees and we show that Gradient Boosted Trees is a better choice than Random Forests when using an entity-dependent approach. While our results with Gradient Boosted Trees show that this method is competitive, it does not achieve good enough results to warrant completely replacing Random Forests as the standard machine-learning algorithm for CCR. However our results indicate that future work on building LTR based CCR systems should not assume the established RankLib + Random Forest implementation is the best off-the-shelf ranking library to use. We also have reason to believe that with good hyper-parameter estimation Gradient Boosted Trees can outperform Random Forests, but experiments into this are currently limited by the lack of a validation set for the TREC KBA track truth data.

RQ3 How does entity-dependent approaches compare to entity-independent approaches for ranking?

Our results show that the choosing an entity-dependent approach can give better results in terms of the F1 measure than using an entity-independent approach. Our results also show that the choice of machine-learning algorithm must be taken into consideration when choosing between an entity-independent vs entity-dependent approach, as the Random Forests algorithm seeming performs better under an entity-independent approach while Gradient Boosted Trees performs better with the entity-dependent approach. The fact that we are unable to create a entity-dependent run which produces statistically significant results over a entity-independent baseline implementing indicates that the choice between entity-dependent or entity-dependent approach should not be a major focus in future research, at least given the current state of the truth data used in the TREC KBA track. In the event that a larger training set becomes available for CCR we believe it would be worth reevaluating the choice of using an entity-dependent vs entity-independent approach.

RQ4 How does the choice of features affect the performance of supervised learning approaches?

Our system performs surprisingly well when using only a simple feature set based on the work presented in Balog Krisztian et al. (2013). By expanding the basic feature set with additional features specifically engineering for aiding the system distinguish between vital and non-vital documents and showing this improves performance across all LTR-based approaches we show that good feature engineering is a consistent way of improving performance of LTR-based CCR systems. We also show that high-dimensional document-representation features can improve model performance in some cases, however the specific way these features are implement is important because some variants we implemented negatively affect model performance.

7.3 Conclusion

In this thesis we have implemented an LTR-based CCR system which produces good results when evaluated against other approaches in the 2014 TREC KBA track. Our implementation of different variants of the LTR-based approach show that there are interesting relationships between the choice of machine-learning algorithm and choice of either a entity-dependent or entity-independent approach, and that some machine-learning algorithms seem to work better with one than the other. Our implementation of several novel features as well as an improved method of creating entity profiles seems to result in an improvement of the performance across all approaches which shows the importance of feature engineering in CCR.

In our discussion we present some of our ideas as to what prevents our solution from getting better results. One recurring trend in the discussion is that many of the limitations and problems when creating CCR systems originate in the current state of StreamCorpus and the associated truth data. We believe based on our findings in this thesis that research on CCR is currently limited by (1) the lack of balanced training sets for all target entities which limits the effectiveness of entity-dependent approaches and (2) the lack of a validation set which effectively limits research on CCR to using relatively simple machine-learning algorithms on their default settings.

There is really only one way to fix this issue, which is to create better truth data with more evenly distributed training data across target entities. Considering there have been no attempts (to our knowledge) at updating the truth data for CCR since the 2014 TREC KBA track, we believe it would be highly beneficial to future research on CCR for someone to create better truth data for CCR with more and better training data.

Once this happens we believe it would be interesting to revisit the research questions posed in this thesis. Given more and better training data we believe we can achieve better results by training better entity-dependent ranking models, and using the parameters of machine-learning algorithms more properly.

7.4 Future work

During our work on this thesis we came up with several ideas which we would like to explore further, but weren't able to due to either time constraints or the idea being too unrelated to the main goals of this thesis. In this section we will briefly present some of these ideas which we believe could be of interest in future research on cumulative citation recommendation.

7.4.1 Investigate effects of hyper-parameters on model performance

As we discussed earlier one of the interesting contributions of this thesis is that using Gradient Boosted Trees in ranking mode via XGBoost performs at least as good if not better than Random Forests using default settings. While the Random Forest algorithm is relatively limited in the choices of hyper-parameters, XGBoost provides several hyper-parameters which could significantly affect model performance. We are especially interested in looking at the effects of tuning the regularization parameter of XGBoost since this is one of the major advantages XGBoost has over similar machine-learning algorithms such as Random Forests. We believe that proper hyper-parameter is much more important for when using XGBoost than Ranklib + Random Forests since there are many more hyper-parameters in XGBoost than Ranklib's Random Forests implementation.

One of the things making hyper-parameter estimation difficult when using the truth data from the 2014 TREC KBA track is that there is no validation set with which to make model choices with. One possible way to resolve this is to create a validation set by splitting the entity-document pairs in the training set in order to create a smaller training set with associated validation set. This is however not trivial because one has to ensure there are enough document-entity pairs in both the training set and validation set, which could be problematic given there is very few document-entity pairs for some of the entities: Splitting these further may results in too few training instances for some entities.

7.4.2 Research the UX aspects of CCR

This thesis as well as most other papers relating to CCR address the task of filtering vital documents from a collection. However to our knowledge little to no research has been done on on the user experience (UX) aspect of CCR or KBA in general. We believe that without a proper front-end system to present the filtered documents, CCR will forever remain a research topic and never be of use outside academia.

For this reason we believe that a natural next step in the evolution of CCR systems is the design and implementation of a front-end for these systems that can present the filtered documents in a way that will allow knowledge base editors to browse the filtered documents and easily integrate the information they contain into existing knowledge bases. Research into this front-end has to find innovative ways to explain to the user *why* each filtered document is relevant, which parts of the document are relevant and which are not, and possibly even improve the system performance by having humans judge the filtered documents by identifying false positives. We propose that a type of CCR based search engine could be developed which allow knowledge base editors to search for new relevant documents to cite for a given entity. This research could take inspiration from existing research on UX for recommender systems such as Knijnenburg et al. (2012).

7.4.3 Create a combined document collection for the 2012-2014 TREC KBA tracks

In all works on the TREC KBA track CCR task we have seen a choice is always made to use only the StreamCorpus collection and associated truth data from one of the 2012, 2013 or 2014 TREC KBA tracks. All of these tracks have different target entities, and therefore disjoint truth data. In order to create a larger collection of truth data it would be possible to combine the target entity list of the 2012-2014 TREC KBA tracks, effectively creating a much larger training and testing set, with considerably more target entities. Since the 2014 StreamCorpus iteration also is a superset of all previous StreamCorpus collections, it would be possible to combine the truth data from the three TREC KBA tracks and simply use the 2014 StreamCorpus. This could make it possible to train more complex models as more training data is available.

This task is however not trivial because of the size of the 2014 StreamCorpus. Almost all participants in the 2014 TREC KBA track use the filtered version of the 2014 StreamCorpus collection, which contains significantly fewer documents, but also cannot be used for the 2012 and 2013 target entities because many documents that mention these entities are filtered out of the collection. Given sufficient computational power, such as with a large cluster computing network it would be possible to download the whole 2014 StreamCorpus and create a new filtered collection that filters out documents mentioning the target entities from all TREC KBA tracks which makes working with combined target entity list feasible for researchers with less available computational power. If this new filtered collection could be made publically available it would also allow other researcher to access this larger training and testing set which could lead to many interesting new approaches to CCR.

Bibliography

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. “Leveraging Linguistic Structure For Open Domain Information Extraction”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. 2015, pp. 344–354.

Ricardo Baeza-Yates A. and Berthier Ribeiro-Neto. *Modern Information Retrieval*. 1999.

Balog Krisztian, Takhirov Naimdjon, Ramampiaro Heri, and Nørvåg Kjetil. “Multi-step Classification Approaches to Cumulative Citation Recommendation”. In: *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*. 2013, pp. 121–128.

Krisztian Balog and Heri Ramampiaro. “Cumulative Citation Recommendation : Classification vs . Ranking”. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2012, pp. 941–944.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. “A Neural Probabilistic Language Model”. In: *The Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.

Richard Berendsen, Edgar Meij, Daan Odijk, Maarten de Rijke, and Wouter Weerkamp. “The University of Amsterdam at TREC 2012”. In: *Proceedings of The Twenty-First Text REtrieval Conference*. 2012.

Ignacio Cano, Sameer Singh, and Carlos Guestrin. “Distributed Non-Parametric Representations for Vital Filtering: UW at TREC KBA 2014”. In: *Proceedings of The Twenty-Third Text REtrieval Conference*. 2014.

Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. “Representation learning for very short texts using weighted word embedding aggregation”. In: *Pattern Recognition Letters* 80 (2016), pp. 150–156.

Laura Dietz, Jeffrey Dalton, and Krisztian Balog. “Time-aware Evaluation of Cumulative Citation Recommendation Systems”. In: *Proceedings of SIGIR 2013 Workshop on Time-aware Information Access*. 2013.

-
- Miles Efron, Jana Deisner, Peter Organisciak, Garrick Sherman, and Ana Lucic. “The University of Illinois’ Graduate School of Library and Information Science at TREC 2012”. In: *Proceedings of The Twenty-First Text REtrieval Conference*. 2012.
- William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.
- John R. Frank, Steven J. Bauer, Max Kleiman-Weiner, Daniel a. Roberts, Nilesh Tripurani, Ce Zhang, Christopher Re, Ellen Voorhees, and Ian Soboroff. “Evaluating Stream Filtering for Entity Profile Updates for TREC 2013”. In: *Proceedings of The Twenty-Second Text REtrieval Conference*. 2013.
- John R Frank, Max Kleiman-Weiner, Daniel A Roberts, Ellen Voorhees, and Ian Soboroff. “Evaluating Stream Filtering for Entity Profile Updates in TREC 2012, 2013, and 2014”. In: *Proceedings of The Twenty-Third Text REtrieval Conference*. 2014.
- Gebrekirstos G Gebremeskel, Jiyin He, Arjen P De Vries, and Jimmy Lin. “Cumulative Citation Recommendation: A Feature-Aware Comparison of Approaches”. In: *25th International Workshop on Database and Expert Systems Applications*. 2014, pp. 193–197.
- Jingtian Jiang, Chin-yew Lin, and Yong Rui. *MSR KMG at TREC 2014 KBA Track Vital Filtering Task*. Tech. rep. 2014.
- Nattiya Kanhabua, Roi Blanco, and Kjetil Nørvåg. “Temporal Information Retrieval”. In: *Foundations and Trends in Information Retrieval*. Vol. 9. 2015, pp. 91–208.
- Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. “Explaining the user experience of recommender systems”. In: *User Modeling and User-Adapted Interaction* 22 (2012), pp. 441–504.
- Matt J Kusner, Yu Sun, Nicholas I Kolkin, and Kilian Q Weinberger. “From Word Embeddings To Document Distances”. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 957–966.
- Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.
- Rachel Tsz-Wai Lo, Ben He, and Iadh Ounis. “Automatically Building a Stopword List for an Information Retrieval System”. In: *JDIM* 3 (2005), pp. 3–8.
- Lerong Ma, Dandan Song, Lejian Liao, and Yao Ni. “A joint deep model of entities and documents for cumulative citation recommendation”. In: *Cluster Computing* (2017).
- Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- Trevor N. Mansuy and Robert J. Hilderman. “Evaluating WordNet Features in Text Classification Models”. In: *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference* (2006), pp. 568–573.
- Jiana Meng, Hongfei Lin, and Yuhai Yu. “A two-stage feature selection method for text categorization”. In: *Computers & Mathematics with Applications* 62 (2011), pp. 2793–2800.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* (Jan. 2013).
- George A. Miller. “WordNet: A Lexical Database for English”. In: *Communications of the ACM* 38 (1995).

Margaret Mitchell. “Overview of the TAC2013 Knowledge Base Population Evaluation: English Sentiment Slot Filling”. In: *Proceedings of the Sixth Text Analysis Conference*. 2013.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

Hung Nguyen, Yi Fang, Sandhya Gade, Vijay Mysore, Juan Hu, Sabita Pandit, Aparna Srinivasan, and Miao Jiang. “A Pattern Matching Approach to Streaming Slot Filling”. In: *Proceedings of The Twenty-Second Text REtrieval Conference*. 2013.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. “GloVe : Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014).

Ridho Reinanda, Edgar Meij, and Maarten de Rijke. “Document Filtering for Long-tail Entities”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 771–780.

S Robertson and I Soboroff. “The TREC 2002 Filtering Track report”. In: *The Eleventh Text REtrieval Conference*. 2002, pp. 27–39.

Hassan Saif, Miriam Fernández, Yulan He, and Harith Alani. “On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter”. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation*. 2014, pp. 810–817.

Tetsuya Sakai. “Statistical Reform in Information Retrieval?” In: *SIGIR Forum* 48 (2014), pp. 3–12.

G. Salton, A. Wong, and C. S. Yang. “A Vector Space Model for Automatic Indexing”. In: *Communications of the ACM* 18 (Nov. 1975), pp. 613–620.

Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. “Open language learning for information extraction”. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 2012, pp. 523–534.

Mark D Smucker, James Allan, and Ben Carterette. “A Comparison of Statistical Significance Tests for Information Retrieval Evaluation”. In: *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. 2007, pp. 623–632.

Alper Kürşat Uysal and Serkan Gunal. “The impact of preprocessing on text classification”. In: *Information Processing & Management* 50 (2014), pp. 104–112.

Ronald E Walpole, Raymond H Myers, Sharon L Myers, and Keying Ye. *Probability and statistics for engineers and scientists*. Ninth Edition. Pearson London, 2012.

Jingang Wang, Lejian Liao, Dandan Song, Lerong Ma, Chin-Yew Lin, and Yong Rui. “Resorting Relevance Evidences to Cumulative Citation Recommendation for Knowledge Base Acceleration”. In: *Web-Age Information Management*. Springer International Publishing, 2015, pp. 169–180.

Jingang Wang, D Song, CY Lin, and L Liao. “BIT and MSRA at TREC KBA CCR Track 2013”. In: *Proceedings of The Twenty-Second Text REtrieval Conference*. 2013.

Jingang Wang, Zhiwei Zhang, Ning Zhang, Dandan Song, and Luo Si. “BIT and Purdue at TREC-KBA-CCR Track”. In: *Proceedings of The Twenty-Third Text REtrieval Conference*. 2014.

Chuan Wu, Wei Lu, Pengcheng Zhou, and Xiaohua Feng. “WHU at TREC KBA Vital Filtering Track 2014”. In: *Proceedings of The Twenty-Third Text REtrieval Conference* (2014).

Per-topic training data distribution

Entity	Non-referent	Unknown	Useful	Vital
Rick_Hansen	2433	0	30	8
Jean_Luc_Bilodeau	0	0	3	2
Randy_Dorn	0	2	157	74
Shelley_Redinger	0	0	0	50
Maelle_Ricker	1	0	54	17
Rose_Egge	0	0	1	1
Jessie_Kaech	0	0	10	3
Nolan_Watson	0	0	14	7
BNSF_Railway	5	11	384	34
Theresa_Spence	0	0	73	63
Peter_Goldmark	19	1	152	27
Robyn_Gervais	0	0	1	33
_Bill_Templeton	0	0	19	5
Jeff_Mangum	1	0	21	20
King_Cat_Theater	2	82	23	36
Missing_Women_				
Commission_of_Inquiry	0	0	37	45
Lisa_Brown	17	0	21	6
Ted_Prior	2	0	1	3
Nordic_Heritage_Museum	0	0	16	21
Paul_Watson	12	1	33	2
Stephen_Buxbaum	0	0	4	4
Jim_Busey	0	0	12	24
Damien_Jurado	0	1	70	13
Josh_Vander_Vies	0	0	0	3
Susan_Chapelle	0	0	5	5
IslandWood	1	274	58	2
Shawn_Atleo	0	0	38	73
Bryce_Leavitt	0	0	56	14

Cameron_Ward	1	3	30	11
Dan_Satterberg	2	62	415	13
Lisa_Muri	0	0	6	3
Lizette_Graden	0	0	0	62
Brock_Schuh	0	0	9	15
Mark_Lindquist	24	33	468	20
Brodie_Clowes	14	0	3	3
Paul_Brandt	0	9	9	13
Dow_Constantine	1	5	488	33
Nancy_Wilhelm_Morden	0	0	3	14
Carmela_Dellino	0	0	25	17
Abbotsford_Arts_Centre	0	0	24	1
Mason_Wilgosh	0	0	18	20
Ralph_Dannenberg	0	0	54	7
Andy_Billig	0	0	25	22
Anne_Blair	3	0	9	8
Marty_McLaren	0	0	27	6
Tulalip	555	3	62	12
Leona_Aglukkaq	0	1	680	33
_3NIbfDpEdTwZ	0	0	7	6
Mike_Kluse	0	0	17	2
Snohomish_High_School	0	1	200	4
Tsawwassen_First_Nation	0	0	22	16
J._Tillman	2	1	46	19
Elmer_Derrick	0	0	3	64
Bryan_Raiser	0	0	3	3
Jonathan_Meline	0	0	1	46
Ted_Sturdevant	0	0	179	17
Spokane_Tribe	0	0	83	19
Spokane_County_Raceway	196	0	70	11
Corisa_Bell	0	0	3	18
Semiahmoo_First_Nation	0	0	4	4
Tsleil-Waututh_First_Nation	0	0	2	2
Genaveve_Starr	0	0	3	3
Chad_Kroeger	1	0	287	115
Georgie_Bright_Kunkel	0	0	8	4
Rob_Kirkham	0	0	3	1
Joby_Shimomura	0	0	21	19
Spokane_Convention_Center	0	0	77	16
matt_manweller	0	0	142	16
Jacob_Hoggard	0	4	38	9
_Dave_Rosin	0	0	0	1
Kshama_Sawant	0	2	229	15
Jesse_Sykes	2	0	21	4
Kalispel_Tribe	0	0	184	8
James_Windle	8	26	181	10

Table A.1: Number of training examples for each relevancy level per entity in the training data of the 2014 TREC KBA track (Target entities of the CCR task only)

Appendix **B**

Command-line used for evaluating runs

```
python -m kba.scorer.ccr  
    --cutoff-step 1  
    --any-up  
    [runfile directory]  
    [annotation file]
```