Feeding the Second Screen: Semantic Linking based on Subtitles

Daan Odijk
d.odijk@uva.nl

Edgar Meij
edgar.meij@uva.nl

Maarten de Rijke
derijke@uva.nl

ISLA, University of Amsterdam
Amsterdam

ABSTRACT

Television is changing. Increasingly, broadcasts are consumed interactively. This allows broadcasters to provide consumers with additional background information that they may bookmark for later consumption. To support this type of functionality, we consider the task of linking a textual streams derived from live broadcasts to Wikipedia. While link generation has received considerable attention in recent years, our task has unique demands that require an approach that needs to (i) be high-precision oriented, (ii) perform in real-time, (iii) work in a streaming setting, and (iv) typically, with a very limited context. We propose a learning to rerank approach that significantly improves over a strong baseline in terms of effectiveness and whose processing time is very short. We extend this approach, leveraging the streaming nature of the textual sources that we link by modeling context as a graph. We show how our graph-based context model further improves effectiveness. For evaluation purposes we create a dataset of segments of television subtitles that we make available to the research community.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

1. INTRODUCTION

Television broadcasts are increasingly consumed on an interactive device or with such a device in the vicinity. Around 70% of tablet and smartphone owners use their devices while watching television [25]. These developments allow the television audience to interact with the content they are consuming, extending the viewing experience both live and on-demand. The interaction includes not only producing and consuming broadcast-specific social media, but it also caters for providing content that is created exclusively for the interactive device, such as additional background information. When an interactive device is used in this fashion, it is commonly referred to as a *second screen*.

For live television, edited broadcast-specific content to be used on second screens is hard to prepare in advance. We present an approach for automatically generating links to background information in real-time, to be used on second screens. Our approach automatically generates links to Wikipedia. This process is commonly known as *semantic linking* and has received much attention in recent years [10, 18, 19, 22, 23]. Such links are typically explanatory, enriching the link source with definitions or background information [4, 13]. Recent work has considered semantic linking for short texts such as queries and microblogs [18–20]. The performance of generic methods for semantic linking deteriorates in such settings, as language usage is creative and context virtually absent.

We base our semantic linking approach for television broadcasts on subtitles, thereby effectively casting the task as one of identifying and generating links for elements in the stream of subtitles. Note that the subtitles are not actually shown, but only used as a textual stream to generate links that may then be shown through a visual representation, on a second screen, as sketched in Fig. 1. Traditional document-based approaches to semantic linking are not suited for this task, as links need to be generated continuously from this stream. On the other hand, using semantic linking approaches for short text, that completely ignore the streaming nature of the material, would mean missing out on important contextual signals. Hence, in order for our semantic linking approach to be effective in the context of second screens, it needs to be fast, able to disambiguate between candidate links in real-time, and leverage streaming data so as to capture context. Furthermore, since the viewer is dividing her attention between the actual broadcast and the second screen, the information that is being offered needs to be of high quality, i.e., have high precision.

We propose a learning to rerank approach to improve upon a strong baseline retrieval model for generating links from streaming text. In addition, we model context using a graph-based approach. This approach is particularly appropriate in our setting as it allows us to combine a number of context-based signals in streaming text and capture the core topics relevant for a broadcast, while allowing real-time updates to reflect the progression of topics being dealt with in the broadcast. Our graph-based context model is highly

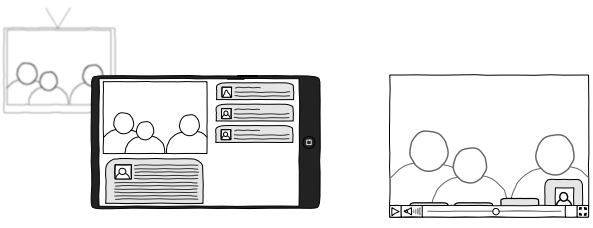

Figure 1: Sketches of a second screen (left) and an interactive video player (right) showing links to background information, synchronized with a television broadcast. Links pop up briefly when relevant and are available for bookmarking or exploring.

OAIR 2013, May 22–24, 2013, Lisbon, Portugal.

accurate, fast, allows us to disambiguate between candidate links and capture the context as it is being built up.

We address the following three research questions:

RQ1 What is the performance, in terms of effectiveness and efficiency, of a state-of-the-art retrieval model on the task of semantic linking of streaming text?

RQ2 Is a learning to rerank approach with task-specific features able to improve the effectiveness over the retrieval baseline, and, if so, at what computational costs?

RQ3 Can we leverage the streaming nature of a textual source, by modeling context as a graph, to improve the effectiveness of semantic linking?

Our main contribution is a set of effective feature-based methods for performing real-time semantic linking. We show how a learning to rerank approach for semantic linking performs on the task of real-time semantic linking, in terms of effectiveness and efficiency. We extend this approach with a graph-based method to keep track of context in a textual stream and show how this can further improve effectiveness. By investigating the effectiveness and efficiency of individual features we provide insight in how to improve effectiveness while maintaining efficiency for this task. Additional contributions include a formulation of a new task: semantic linking of a textual stream, and the release of a dataset for this new task, including ground truth.

2. TASK DEFINITION

The task we describe is real-time semantic linking of a textual stream. We link subtitles that come with television broadcasts—live or recorded—to Wikipedia articles. The identified links should be interesting and relevant for a wide audience. In the context of subtitles, we define a dynamic textual stream as a source that continually produces "chunks" of text. A *chunk* is the amount of subtitle text that can be displayed on a single screen (again, we are not assuming that the text is actually displayed, only that it comes with the broadcast). Chunks are relatively short and contain approximately seven terms on average. Therefore, chunks do not necessarily form a grammatical sentence. However, as these chunks are produced to be read in sequence, syntactic phrases generally do not cross chunk boundaries. Chunks form a growing sequence $T = \langle t_1, \ldots \rangle$ and our task is to decide, in real-time, whether a link to Wikipedia should be created for t_i and what the link target should be.

A *link candidate* $l_i \in L$ links an "anchor" a in chunk t_i to a target w; an *anchor* is a term n-gram within in a chunk. Each target w is a Wikipedia article proper, i.e., excluding redirect and disambiguation pages, from a set of Wikipedia articles W. A target is identified by its unique title on Wikipedia. In the dataset we manually identified video segments of a television program; here, a video segment is a set of chunks that share a single topic, for example an interview with a guest in a talk show.

3. REAL-TIME SEMANTIC LINKING

Next, we introduce our approach to real-time semantic linking. It consists of a baseline retrieval model that is based on how links between Wikipedia articles are created. We improve over this baseline using a learning to rerank approach using a set of lightweight features. Finally, we describe an extension to the learning to rerank approach that explicitly models context.

3.1 Baseline retrieval model

Our method for real-time link generation consists of three steps: link candidate finding, ranking and reranking. We consider the first two steps our baseline retrieval model. In this model, each Wikipedia article is represented by the anchors that are used to link to it within Wikipedia. The first, recall-oriented step is aimed at finding as many link candidates as possible. Here, we produce a set of link candidates L for each chunk t_i that each link to a Wikipedia article w. To this end, we perform lexical matching of each n-gram a of chunk t_i with the anchor texts found in Wikipedia.

The second step is to rank the link candidates in L. In particular, we can use statistics on the anchor text usage. We consider the prior probability that anchor text a links to Wikipedia article w:

$$COMMONNESS(a, w) = \frac{|L_{a,w}|}{\sum_{w' \in W} |L_{a,w'}|}, \qquad (1)$$

where $L_{a,w}$ denotes the set of all links with anchor text a and target w. The intuition is that link candidates with anchors that always link to the same target are more likely to be a correct representation than those for which the anchor text is used more often to link to other targets.

3.2 Learning to rerank

The third step is aimed at improving precision using a learning to rerank approach, that was effective on similar tasks [16, 20, 23]. For link candidates many ranking criteria are in play, making learning to rerank particularly appropriate. We use a set of lightweight features, that can be computed online. We use variants of features proposed by Meij et al. [20], that are suited for our television broadcast context. These 26 features are listed in Table 1. $WIKISTATS_n(w)$ is an indication of the popularity of a Wikipedia article and is defined as the absolute number of visitors for an article w in the n days before the television broadcast.[1] $WIKISTATSTREND_{n,m}(w)$ is the relative number of visitors in the last n days, compared to the number of visitors in the last m days. This feature is intended to provide information on peaks in the number of visitors. The intuition is that these features help to identify popular current topics in the television broadcast. We use a decision tree based approach as it has outperformed Naive Bayes and Support Vector Machines on similar tasks [20, 23]. We choose Random Forests [2] as it is robust, efficient and easily parallelizable.

3.3 Modeling context

Link generation methods that rely on an entire document are not suited for use in the context of streaming text. Such methods are computationally expensive, due to the many comparisons that have to be made, as explained below in §6. Since we want to generate links for each chunk, we would need to do these comparisons for each chunk in the stream. What we need, instead, is a method to model context that can be incrementally updated and by which we can easily compute features for link candidates. We model the context of a textual stream as an undirected graph as follows. A *context graph* $G = (V, E)$, comprises a set V of vertices and a set E of edges; vertices are either a chunk t_i, a target w or an anchor (t_i, a). Edges link each chunk t_i to t_{i-1}. Furthermore, for each anchor (t_i, a), there is an edge from (t_i, a) to t_i and one from (t_i, a) to w. The graph reflects the content of the textual stream and encodes the structure by connecting chunks. This results in a smaller distance for things that are mentioned together. Furthermore, nodes for Wikipedia articles that are mentioned more often,

[1] See `http://dumps.wikimedia.org/other/pagecounts-raw/`

Table 1: Features used for the learning to rerank approach.

Anchor features	
$LEN(a) = \|a\|$	Number of terms in the n-gram a
$IDF_f(a)$	Inverse document frequency of a in representation f, where $f \in \{\text{title}, \text{anchor}, \text{content}\}$
$KEYPHRASE(a)$	Probability that a is used as an anchor text in Wikipedia (documents)
$LINKPROB(a)$	Probability that a is used as an anchor text in Wikipedia (occurrences)
$SNIL(a)$	Number of articles whose title equals a constituent n-gram of a
$SNCL(a)$	Number of articles whose title match a constituent n-gram of a
Target features	
$LINKS_f(w)$	Number of Wikipedia articles linking to or from w, where $f \in \{\text{in}, \text{out}\}$ respectively
$GEN(w)$	Depth of w in Wikipedia category hierarchy
$REDIRECT(w)$	Number of redirect pages linking to w
$WIKISTATS_n(w)$	Number of times w was visited in the last $n \in \{7, 28, 365\}$ days
$WIKISTATSTREND_{n,m}(w)$	Number of times w was visited in the last n days divided by the number of times w visited in last m days, where the pair $(n, m) \in \{(1, 7), (7, 28), (28, 365)\}$
Anchor + Target features	
$TF_f(a, w) = \frac{n_f(a,w)}{\|f\|}$	Relative phrase frequency of a in representation f of w, normalized by length of f, where $f \in \{\text{title}, \text{first sentence}, \text{first paragraph}\}$
$POS_1(a, w) = \frac{pos_1(a)}{\|w\|}$	Position of first occurrence of a in w, normalized by length of w
$NCT(a, w)$	Does a contain the title of w?
$TCN(a, w)$	Does the title of w contain a?
$TEN(a, w)$	Does the title of w equal a?
$COMMONNESS(a, w)$	Probability of w being the target of a link with anchor text a

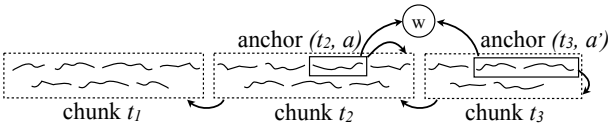

Figure 2: Illustration of a context graph with three chunks (t_1, t_2, t_3) and two link candidates with different anchors $((t_2, a)$ and $(t_3, a'))$, but the same target w.

will have more anchors connecting to them, making these nodes more central and thereby more important in the graph. Fig. 2 shows an illustration of a context graph with three chunks.

Constructing the context graph. At the start of each video segment we initialize an empty context graph. The algorithm that describes how we update the context graph is described in Algorithm 1. We do not add vertices and edges to the context graph for every link candidate, for two reasons. First, the algorithm to find link candidates is recall-oriented and, therefore, produces many links for each chunk; a single usage of an anchor for a Wikipedia article is enough for it to be considered a link candidate. This introduces links that are very unlikely or trivial and can be considered noise. Second, given our objective to perform semantic linking in real time, we need to limit the size of the context graph.

We select what link candidates to add to the context graph by placing a threshold, τ, on the probability of a link candidate being correct. The baseline ranking function $COMMONNESS(a)$ as defined in Eq. 1, can be seen as an estimate for this probability for a Wikipedia article. The probability that an n-gram a is used as an anchor is estimated by the feature

$$LINKPROB(a) = \frac{\sum_{w \in W} |L_{a,w}|}{\sum_{w' \in W} n(a, w')}, \tag{2}$$

Algorithm 1: Updating the context graph.

Data: A context graph $G = (V, E)$, comprising a set V of vertices or nodes and a set E of edges.
On initialization, $V = \emptyset, E = \emptyset$
Input: Set of link candidates $L = \{l_1, ..., l_M\}$ that each link a specific anchor (t_i, a) to a Wikipedia article $w \in W$ for chunk t_i
Input: A ranking function r and a threshold τ
Result: Nodes and edges added to G for link candidates in L if $r(a, w) > \tau$

$V \longleftarrow V \cup t_i$;
if $i > 0$ **then**
  $E \longleftarrow E \cup \{t_i, t_{i-1}\}$;
foreach $l = ((t_i, a), w) \in L$ **do**
  if $r(a, w) > \tau$ **then**
    $V \longleftarrow V \cup (t_i, a) \cup w$;
    $E \longleftarrow E \cup \{(t_i, a), t_i\} \cup \{(t_i, a), w\}$;

where $L_{a,w}$ denotes the set of all links with anchor text a and target w and $n(a, w)$ is the number of occurrences of n-gram a in w. If we combine these two probabilities, we have an estimate for the probability that an n-gram a is used as an anchor linking to Wikipedia article w. We define this combined probability as

$$SENSEPROB(a, w) = \frac{|L_{a,w}|}{\sum_{w' \in W} n(a, w')}, \tag{3}$$

where $L_{a,w}$ and $n(a, w)$ denote the same as in Eq. 2. We use $SENSEPROB$ to select candidates to add to a context graph, as it captures both whether a link is correctly disambiguated and the likelihood of the anchor text being a link.

We set the threshold value so that we can easily update the graph and compute the features below during the time a particular sub-

Table 2: Context features used for learning to rerank on top of the features listed in Table 1.

Context features	
$DEGREE(w, G)$	Number of edges connected to the node representing Wikipedia article w in context graph G.
$DEGREE-CENTRALITY(w, G)$	Centrality of Wikipedia article w in context graph G, computed as the ratio of edges connected to the node representing w in G.
$PAGERANK(w, G)$	Importance of the node representing w in context graph G, measured using PageRank.

title is shown on the screen (roughly four seconds). On a small development set, this resulted in $\tau = 0.1$. We prune the graph for nodes that were added more than 100 chunks ago (∼7 minutes on our dataset, see below).

Computing features from context graphs. To feed our learning to rerank approach with information from the context graph we compute a number of features for each link candidate. These features are described in Table 2. First, we compute the degree of the target Wikipedia article in this graph. To measure how closely connected a target is, we compute degree centrality. Finally, we measure the importance of a target by computing its PageRank [26].

4. EXPERIMENTAL SETUP

We describe the dataset used, our ground truth and metrics.

Dataset. To measure the effectiveness and efficiency of our proposed approach to semantic linking, we use the subtitles of six episodes of a live daily talk show. The subtitles are generated during live broadcast by a professional and are intended for the hearing impaired. From these subtitles, video segments are identified, each covering a single item of the talk show. These video segments are based on the structure of the talk show. Video segments cover a single topic; their boundaries are manually identified during annotation. We leave automatically identifying video segment boundaries for future work. Our data set consists of 5,173 chunks in 50 video segments, with 6.97 terms per chunk. The broadcast time of all video segments combined is 6 hours, 3 minutes and 41 seconds.

Establishing ground truth. In order to train the supervised machine learning methods and also evaluate the end result, we need to establish a gold standard. To this end, we have asked a trained human annotator to manually identify links that are relevant for a wide audience.[2] The subtitles are in Dutch, so we link to a Dutch version of Wikipedia.[3] Each video segment is assessed in sequence and links are identified by selecting anchors and a target Wikipedia article. If no target can be identified a link with a NIL target is created. A total of 1,596 links have been identified, 150 with a NIL target and 1,446 with a target Wikipedia article, linking to 897 unique articles, around 17.94 unique articles per video segment and 2.47 unique articles per minute.

Evaluation metrics. For the evaluation of our learning to rerank approach, we are specifically interested in the ranking that is assigned to each link. We therefore regard the ranked list of all target Wikipedia articles for the link candidates that are produced by the baseline retrieval model for a video segment. Our learning to rerank approach assigns new ranks for the ranked list, but does not update the elements making up the list. We report average R-precision and mean average precision (MAP). We also measure efficiency. We report the average classification time per chunk on a single core of an eight core machine. This classification time per chunk indicates how long it takes to produce links for one line of subtitles, after they appear on the screen. It should be noted, that *all* features can be computed off-line, only requiring a simple lookup at runtime.

Features and baselines. To compute our features, we use a Wikipedia dump from November 2011 (∼1M articles) for which we calculate link statistics. For the $WIKISTATS$ features, we collect visitor statistics for Wikipedia on a daily basis and aggregate these per day, week, month, and year. This preprocessing makes all features fairly easy to compute at runtime. We consider one baseline: a baseline retrieval model using *COMMONNESS* as a ranking function. On top of that we consider a learning to rerank approach as described as step 3 in §3. In this learning to rerank approach, the ranker is trained using five-fold cross-validation at the video segment level. The Random Forests algorithm has two free parameters; the number of trees is set to 1500, based on preliminary experiments reported in §5 and we set K, the number of features to consider when taking a bootstrap sample, according to the common rule-of-thumb, to roughly 10% of the number of features [24].

5. RESULTS AND DISCUSSION

We discuss the outcomes of our experiments aimed at answering our research questions.

Baseline retrieval model. First, we consider the performance of our baseline retrieval model. Line 1 in Table 3 shows the scores for the ranking baseline. The recall oriented link candidate finding step described in §3 produces 120,223 links with 42,265 target articles, including 771 known targets that are in the ground truth (a recall of 0.8595). With this number of link candidates, there is a clear need for ranking. The ranking baseline achieves reasonable effectiveness scores; in absolute terms they are higher than the numbers reported for the $COMMONNESS$ ranking baseline on microblog data in [20]. Recall is comparable to the numbers reported for the baseline on Wikipedia articles in [10, 23], where R-Precision and MAP are not reported, however. As to our first research question, the state of the art retrieval model that we use performs well on the task of semantic linking of streaming text in terms of efficiency. The performance in terms of effectiveness is strong, with high recall-scores. In terms of precision these numbers are comparable to the literature, while leaving room for improvement.

Learning to rerank. Next we consider the performance of the learning to rerank approach (§3.2). Lines 2–10 in Table 3 show the results with a growing set of features. We add groups of features in a coordinate ascent [21], i.e., adding the best feature at each step plus any related feature we can compute without additional costs in classification time. We see that learning to rerank significantly boosts effectiveness over the retrieval baseline, while keeping the average classification time per chunk at around 100 milliseconds.

[2]To validate these manual annotations, we have asked a second annotator to annotate six video segments; 95,9% of links identified by the main annotator were also found by the second one.

[3]There is nothing in our approach, however, that is language specific. One could even argue that semantic linking for Dutch is more difficult than for English as the Dutch version of Wikipedia contains fewer Wikipedia articles.

Table 3: Semantic linking results ranked by classification time. Significant differences, tested using a two-tailed paired t-test, are indicated for lines 2–10 tested against line 1 at $p < 0.05$ ($^{\triangle}$) or $p < 0.01$ ($^{\blacktriangle}$).

	Average classification time per chunk (in ms)	R-Prec	MAP
1. Baseline retrieval model	54	0.6052	0.6463
Learning to rerank approach (listing the features used)			
2. *COMMONNESS* + *LINKPROB* + IDF_{anchor} + *NCT* + *TCN* + *TEN*	92	0.6424	$0.7213^{\triangle}$
3. + *KEYPHRASE*	92	0.6617	$0.7355^{\blacktriangle}$
4. + *REDIRECT* + $LINKS_{in}$ + $LINKS_{out}$	93	$0.6910^{\triangle}$	$0.7624^{\blacktriangle}$
5. + POS_1 + TF_{title} + $TF_{sentence}$ + $TF_{paragraph}$	96	$0.7003^{\triangle}$	$0.7729^{\blacktriangle}$
6. + $IDF_{content}$ + IDF_{title}	97	$0.7057^{\blacktriangle}$	$0.7728^{\blacktriangle}$
7. + *LEN* + *SNCL* + *SNIL*	97	$0.7118^{\blacktriangle}$	$0.7758^{\blacktriangle}$
8. + *GEN*	98	$0.7112^{\blacktriangle}$	$0.7749^{\blacktriangle}$
9. + $WIKISTATS_7$ + ...$STATS_{28}$ + ...$STATS_{365}$	99	$0.7175^{\blacktriangle}$	$0.7849^{\blacktriangle}$
10. + $WIKISTATSTREND_{1,7}$ + ...$TREND_{7,28}$ + ...$TREND_{28,365}$	99	$0.7177^{\blacktriangle}$	$0.7884^{\blacktriangle}$

This makes classification in a streaming text setting almost real-time. Lines 9 and 10 in the table concern the *WIKISTATS* features: while they may gain slightly in terms of R-Precision and MAP, this comes at a big increase in pre-processing effort, as we need to collect visitor statistics for Wikipedia on a daily basis. In our experiments, we refer to line 10 as *the learning to rerank approach*.

Our second research question is whether a learning to rerank approach would be able to outperform our retrieval baseline on the task of semantic linking of streaming text. The results for our learning to rerank approach show that it can be highly effective and significantly improve over the retrieval baseline. We can achieve this high effectiveness at an average online classification time of less than 100 milliseconds, making the learning to rerank approach efficient and suited for usage in real time.

As an aside, we analyze the influence of one parameter for the Random Forest algorithm, the number of trees. We evaluate the effectiveness by taking the average of five runs with all features in Table 1. Fig. 3 shows the results. The effectiveness increases as the number of trees increases and reaches a plateau at a value of about 1000, indicating that a value of 1500 is a safe setting.

Modeling context as a graph. We turn to our third research question, and determine whether modeling context as a graph improves effectiveness on the semantic linking task. We evaluate three features for the context graph (listed in Table 2). The results for the learning to rerank runs with these features added are listed in Table 4. Compared to the learning to rerank approach in line 2, we are able to achieve significantly higher performance. All three features improve effectiveness, with *DEGREECENTRALITY* achieving the best score for average R-Precision and *DEGREE* the best for MAP. The fact that we improve on both measures (R-Prec and MAP) indicates that both recall and precision are improved, by each of the three context graph features. The combination of the three context features does not yield further improvements in effectiveness over the individual context graph features (as shown in line 6). To investigate why the combination does not improve effectiveness, we look at the result of a hypothetical oracle that picks the best of the result for each video segment. This gives us a ceiling value for effectiveness that is very close to the value achieved by the individual feature runs, suggesting that the three features measure qualitatively similar things in the context graph. To verify this, we analyze the difference in effectiveness per video segment for each of the three context features between the learning to rerank approach and its extensions with graph-based contextual information. This difference is plotted in Fig. 4. Comparing the individual context features, we see only minor differences in performance per segment. Combining these features can hardly lead to further improved performance.

To further investigate where the inclusion of contextual information helps and where it hurts, we analyze the difference in effectiveness per video segment between the learning to rerank approach and its extensions with the graph-based context model. Out of the 50 video segments, 37 show an increase and 13 a decrease in effectiveness (for each of lines 2–5 in Table 4). We plot the difference in effectiveness against the number of target Wikipedia articles per video segment in Fig. 5. We observe that for video segments with more links, the extended approach with our graph-based

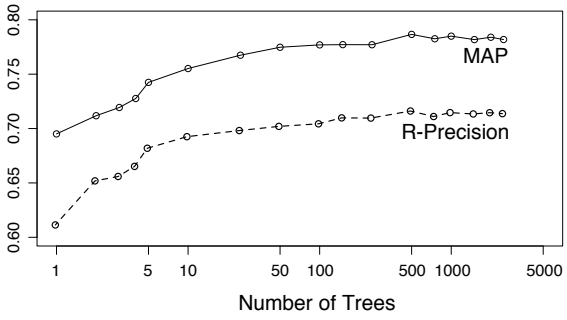

Figure 3: Analyzing the influence of the number of trees, measured in terms of R-precision and MAP.

Table 4: Semantic linking results for the graph-based context model and an oracle run (indicating a ceiling). Significant differences, tested using a two-tailed paired t-test, are indicated for lines 2–6 with $^{-}$ (none), $^{\triangle}$ ($p < 0.05$) and $^{\blacktriangle}$ ($p < 0.01$); the position of the symbol indicates whether the comparison is against line 1 (left most) or line 2 (right most).

	Average classification time per chunk (in ms)	R-Prec	MAP
1. Baseline retrieval model	54	0.5753	0.6235
2. Learning to rerank approach	99	$0.7177^{\blacktriangle}$	$0.7884^{\blacktriangle}$
Learning to rerank (L2R) + one context graph feature			
3. L2R+*DEGREE*	104	$0.7375^{\blacktriangle\triangle}$	$0.8252^{\blacktriangle\blacktriangle}$
4. L2R+*DEGREECENTRALITY*	108	$0.7454^{\blacktriangle\blacktriangle}$	$0.8219^{\blacktriangle\blacktriangle}$
5. L2R+*PAGERANK*	119	$0.7380^{\blacktriangle\triangle}$	$0.8187^{\blacktriangle\blacktriangle}$
Learning to rerank (L2R) + three context graph features			
6. L2R+*DEGREE*+*PAGERANK* +*DEGREECENTRALITY*	120	$0.7341^{\blacktriangle-}$	$0.8204^{\blacktriangle\blacktriangle}$
7. Oracle picking the best out of lines 3–5 for each video segment		*0.7636*	*0.8400*

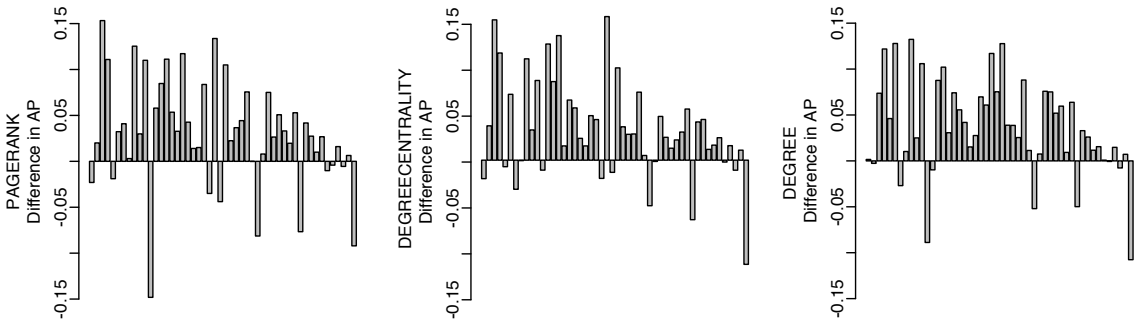

Figure 4: Analyzing the difference in effectiveness per video segment for each context feature, between the learning to rerank approach and the graph-based context model. Effectiveness is measured by AP, for R-precision we observe qualitative similar patterns. Bars are sorted from left to right in increasing performance for the learning to rerank approach.

model consistently improves effectiveness. This indicates that having more context to work with improves the effectiveness of the graph-based context model. On the other hand, we can also observe improvements in the video segments with fewer targets. This indicates that the graph-based context model is also able to improve when context is limited.

In Fig. 5, two video segments stand out by showing relatively low effectiveness scores for both approaches (R-precision below 0.4). Neither video segment contains many links. Looking at their content, these are understandably hard cases as they discuss rare or even obscure news events (one covers the ups and downs of a homeless person, the other a less popular sports contest). Finally, most video segments that show a decrease in effectiveness under the graph-based context model already have a relatively high effectiveness in the learning to rerank approach; the highest decrease was for a video segment with a perfect score.

In terms of efficiency, the $PAGERANK$ feature is computationally more expensive than the other two, but classification time is still low enough for this approach to be used in a real-time setting. Interestingly, the relatively simple to compute $DEGREE$ feature performs relatively well.

Our third research question concerns whether we can improve the effectiveness of our learning to rerank approach if we model context as a graph, leveraging the streaming nature of a textual source. The results in Table 4 show that we can significantly improve the performance of our learning to rerank approach, while maintaining efficiency at an acceptable level, if we add features that were computed from the graph-based context model.

To add or not to add?. The influence of the decision on what links to include in the context graph, as described in §3.3, deserves further attention. First, we use a threshold for what to add to the graph and second, we prune the graph based on age (defined as the n most recent chunks included in the context graph). We consider the effectiveness of three threshold functions: $SENSEPROB$, $COMMONNESS$, $LINKPROB$, with threshold values $\tau \in [0, 0.5]$. Furthermore, we consider five values for the age n: 10, 25, 50, 75 and 100.

Figure 6 shows the results. For the $COMMONNESS$ threshold function (Eq. 1), increasing τ improves results until $\tau = 0.4$. Intuitively this makes sense, as more ambiguous link candidates have lower $COMMONNESS$ values. For the $LINKPROB$ (Eq. 2) and $SENSEPROB$ (Eq. 3) threshold functions, there seems to be a sweet spot for τ between 0.1 and 0.2. The influence of the age value n seems to be less substantial, but there is indication that the higher the value, the better the performance is. The $SENSEPROB$ threshold function we proposed is clearly the most effective of the three functions. Hence, our choice of $\tau = 0.1$ for $SENSEPROB$, with $n = 100$ is a good choice.

6. RELATED WORK

Automatically generating hypertext links has been studied for nearly two decades. Early work included defining a taxonomy of hyperlink types and applying string-matching methods for automatic links [1]. An issue that was raised early on was inter-linker consistency [8]. A typical use case is in linking news archives [12], more recently also across modalities [4]. Henzinger et al. [14] proposed an approach to find relevant news article stories during broadcast news. Every 15 seconds, they produce a two term query to search a news archive, using a 'history feature', containing the last three blocks of text.

Using automatic speech recognition (ASR) for television broadcasts will lead to noisy transcripts, making it difficult to do semantic linking on ASR transcripts. In [5] such transcripts are analyzed in real time to produce entities and topics. VideoCLEF'09 [16] featured a linking task aimed at linking video content to related resources across languages. This task was framed as a known-item-task, where noisy ASR for Dutch was used to produce links to target English Wikipedia articles. The best performance was achieved by taking an off-the-shelf semantic linking toolkit for documents [23]. The learning to rerank approach employed by the toolkit outperformed several retrieval approaches.

Early work on semantic linking, i.e., automatic link generation for Wikipedia, aimed to improve linking on Wikipedia, by finding missing links [11]. Fissaha Adafre and de Rijke clustered similar pages and identified candidate links to be added to a page. These methods to generate links to Wikipedia are also effective outside the Wikipedia domain as several approaches to semantic linking for documents have shown [6, 7, 22, 23, 27]. In this case, links are intended to be explanatory, by providing definitions or background information. In some work, links are created as an concept or entity normalization step [6, 7, 22, 27]. These document-based approaches generally consist of three steps. First, find all candidates for linking and then disambiguate possible targets. From this, select which candidates to link and pick the target to link to. These methods assume text is more or less clean and grammatically correct and rely heavily on context for link selection and disambiguation. This context has been modeled in different ways. Early work uses only local approaches to disambiguation [6, 22], looking at how well each link candidate fits in the text. This is done by comparing the content of the source document to the content the page to be linked to. Global approaches regard all link candidates for a document and then try to form a coherent set from these. These

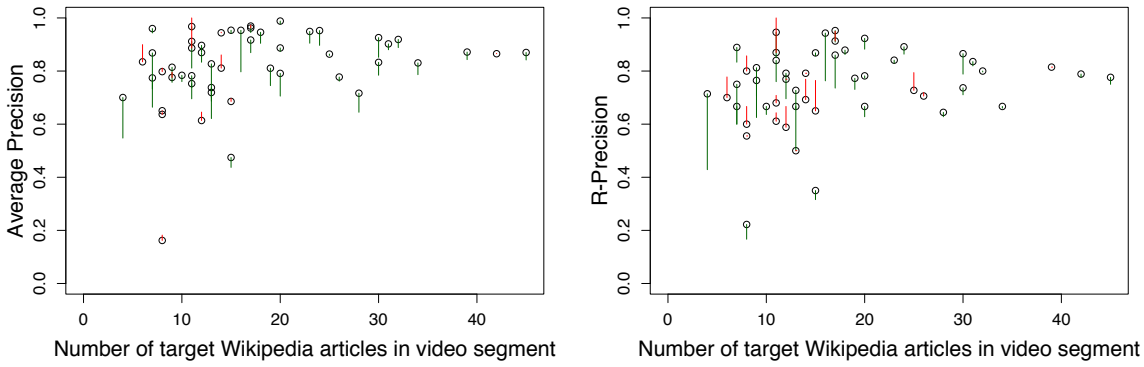

Figure 5: Analyzing the difference in effectiveness in terms of AP (left) and R-Precision (right) per video segment, between the learning to rerank approach and the graph-based context model (using $DEGREECENTRALITY$). For each video segment, a line starts at the effectiveness value for the learning to rerank approach, leading to a circle indicating the effectiveness for the graph-based context model approach.

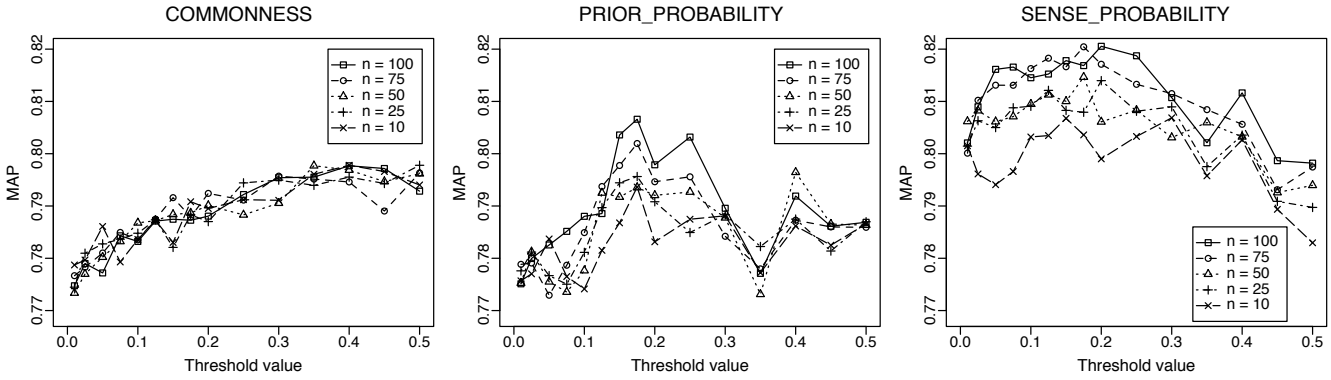

Figure 6: Analyzing the influence of the threshold settings for context graphs for the n most recent chunks, measured by MAP (for R-precision we observe qualitatively similar patterns).

methods use relatedness measures based on the structure of Wikipedia [7, 23, 27]. Milne and Witten [23] balance a measure for how common a link is with how related it is to other links. Comparing all possible links for a document is computationally expensive, so a choice of what link candidates to consider is often made. Using only the few unambiguous link candidates such as in [23] dismisses many link candidates, while considering all makes the set subject to noise and impractically large [7]. For global approaches, semantic relatedness between articles is computed using a distance measure for the incoming and outgoing links and categories an article belongs to [7, 23, 27]. For streaming text, neither local nor global approaches for disambiguation are suited as linking is considered for just a chunk of text. These local and global approaches are computationally heavy, as there are many comparisons to be made for each link candidate.

Semantic linking has been applied to many different specific domains. Jijkoun et al. [15] studied this as an entity normalization task in blogs and comments. He et al. [13] showed that applying semantic linking to radiology reports did not yield satisfactory results and propose a sequential labeling approach, with syntactic features. More recently, work has gone into applying semantic linking to short texts, such as queries [10, 18] and microblogs [20]. In these short texts, generic methods developed for documents fail, as grammar and context are virtually absent. Ferragina and Scaiella [10] developed a system called TAGME, designed specifically for short snippets of poorly composed text. For this they try to find collective agreement for the link targets using voting scheme based on a relatedness score. The authors point out that computing this relatedness score is the most time consuming step in their system. For semantic linking of short texts, Meij et al. [20] have shown that, as a retrieval model for finding link candidates, lexical matching on anchor text performs better than lexical matching on title, language modeling and a document retrieval-based approach.

We have captured context chunks in a structured manner using graphs. Graph-based methods have been proposed for natural language processing (NLP) problems, such as word clustering [3], word dependency [28], text summarization [9] and topic modeling [17]. Erkan and Radev [9] show how a random walk on sentence-based graphs can help in text summarization. A well-known example of this idea of a random walk is PageRank [26]—one of the measures that we use.

7. CONCLUSION

Motivated by the rise in so-called second screen applications we introduced a new task: real-time semantic linking of streaming text. We have created a dataset for this task.[4] We have shown that learn-

[4]The dataset (described in §4) is made available to the research community; it consists of more than 1,500 manually annotated links in over 5,000 subtitle chunk for 50 video segments. See: `http://ilps.science.uva.nl/resource/oair-2013-feeding-second-screen`

ing to rerank can be applied to significantly improve an already competitive retrieval baseline and that this can be done in real-time.

Additionally, we have shown that by modeling context as a graph we can significantly improve the effectiveness of this learning to rerank approach. This graph-based method to keep track of context is especially well-suited for the streaming text, as we can incrementally update the context model.

As to future work, we have shown that selecting what candidate links to add to the graph is an important choice. An interesting follow-up to this observation is to semantically enrich the graph, by weighting the edges of the graph, accounting for the quality of the evidence collected. We can further extend the enrichment to encode more information, e.g., by using the Wikipedia link structure.

The dataset used in this study consists of subtitles from video segments of talk shows. We have chosen talk shows because they cover a range of topics, mostly current. However, our approach is not specific for talk shows and it will be interesting to evaluate this approach on different types of television broadcast, such as live events and sports. Furthermore, with advances in automatic speech recognition (ASR), combining our approach with the output of an ASR-system may provide an effective solution when manually created subtitles are not available.

ACKNOWLEDGEMENTS

This research was partially supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.004.802, 727.011.005, 612.001.116, HOR-11-10, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the BILAND project funded by the CLARIN-nl program, the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), and the Netherlands eScience Center under project number 027.012.105.

REFERENCES

[1] J. Allan. *Automatic hypertext construction*. PhD thesis, Cornell University, 1995.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] C. Brew and S. Schulte im Walde. Spectral clustering for German verbs. In *EMNLP '02*, pages 117–124. ACL, 2002.

[4] M. Bron, B. Huurnink, and M. de Rijke. Linking archives using document enrichment and term selection. In *TPDL '11*, pages 360–371. Springer, 2011.

[5] E. Brown, S. Srinivasan, A. Coden, D. Ponceleon, J. Cooper, A. Amir, and J. Pieper. Towards speech as a knowledge resource. In *CIKM '01*, pages 526–528. ACM, 2001.

[6] R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL '06*, pages 9–16. ACL, 2006.

[7] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP '07*, pages 708–716. ACL, 2007.

[8] D. Ellis, J. Furner-Hines, and P. Willett. On the creation of hypertext links in full-text documents: Measurement of inter-linker consistency. *J. Documentation*, 50(2):67–98, 1994.

[9] G. Erkan and D. Radev. LexRank: Graph-based lexical centrality as salience in text summarization. *J. Artificial Intelligence Research*, 22:457–479, 2004.

[10] P. Ferragina and U. Scaiella. TAGME: On-the-fly annotation of short text fragments (by Wikipedia entities). In *CIKM '10*, pages 1625–1628. ACM, 2010.

[11] S. Fissaha Adafre and M. de Rijke. Discovering missing links in Wikipedia. In *3rd international workshop on Link discovery*, pages 90–97. ACM, 2005.

[12] S. Green. Building hypertext links by computing semantic similarity. *IEEE Trans. on Knowledge and Data Engineering*, 11(5):713–730, 1999.

[13] J. He, M. de Rijke, M. Sevenster, R. van Ommering, and Y. Qian. Automatic link generation with Wikipedia: A case study in annotating radiology reports. In *CIKM '11*, pages 1867–1876. ACM, 2011.

[14] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. *World Wide Web*, 8(2):101–126, June 2005.

[15] V. Jijkoun, M. Khalid, M. Marx, and M. de Rijke. Named entity normalization in user generated content. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 23–30. ACM, 2008.

[16] M. Larson, E. Newman, and G. Jones. Overview of VideoCLEF 2009. In *CLEF '09*, pages 354–368. Springer, 2010.

[17] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *WWW '08*, pages 101–110. ACM, 2008.

[18] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. de Rijke. Learning semantic query suggestions. In *ISWC '09*, pages 424–440. Springer, 2009.

[19] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. de Rijke. Mapping queries to the linking open data cloud: A case study using DBpedia. *J. Web Semantics*, 9(4):418–433, 2011.

[20] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM 2012*, pages 563–572. ACM, 2012.

[21] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[22] R. Mihalcea and A. Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *CIKM '07*, pages 233–242. ACM, 2007.

[23] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *CIKM '08*, pages 509–518. ACM, 2008.

[24] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *JMLR: Workshop and Conference Proceedings*, 14:77–89, 2011.

[25] Nielsen. In the U.S., tablets are TV buddies while ereaders make great bedfellows, May 2012. `http://bit.ly/L4lf9E` [Online; accessed May 2012].

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[27] L. Ratinov, D. Downey, M. Anderson, and D. Roth. Local and global algorithms for disambiguation to Wikipedia. In *ACL '11*, pages 1375–1384. ACL, 2011.

[28] K. Toutanova, C. Manning, and A. Ng. Learning random walk models for inducing word dependency distributions. In *ICML '04*, pages 103–110. ACM, 2004.