Gem-based Entity-Knowledge Maintenance

Bilyana Taneva
Max-Planck Institute for Informatics
Saarbrücken, Germany
btaneva@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Knowledge bases about entities have become a vital asset for Web search, recommendations, and analytics. Examples are Freebase being the core of the Google Knowledge Graph and the use of Wikipedia for distant supervision in numerous IR and NLP tasks. However, maintaining the knowledge about not so prominent entities in the long tail is often a bottleneck as human contributors face the tedious task of continuously identifying and reading relevant sources. To overcome this limitation and accelerate the maintenance of knowledge bases, we propose an approach that automatically extracts, from the Web, key contents for given input entities.

Our method, called GEM, generates salient contents about a given entity, using minimal assumptions about the underlying sources, while meeting the constraint that the user is willing to read only a certain amount of information. Salient content pieces have variable length and are computed using a budget-constrained optimization problem which decides upon which sub-pieces of an input text should be selected for the final result. GEM can be applied to a variety of knowledge-gathering settings including news streams and speech input from videos. Our experimental studies show the viability of the approach, and demonstrate improvements over various baselines, in terms of precision and recall.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

knowledge maintenance; long-tail entities; knowledge acceleration; relatedness; novelty; emerging entities

1. INTRODUCTION

1.1 Motivation

Knowledge bases such as DBpedia (`dbpedia.org`), Freebase (`freebase.com`), or Yago (`yago-knowledge.org`) have become essential assets for Web search, recommendations, analytics, and more. For example, the Google Knowledge Graph is centered around Freebase, and Wikipedia, from which many knowledge bases are derived, has been used for distant supervision in numerous tasks in IR and NLP. While knowledge bases are up-to-date on prominent entities, their maintenance on entities in the long tail and the acquisition of knowledge about newly emerging entities are bottlenecks. The root cause here is the human contributors who need to continuously identify and read relevant sources, in order to update articles or structured knowledge (infoboxes, categories, etc.) on long-tail or emerging entities.

New articles in Wikipedia are first created as stub pages which lack the desired encyclopedic coverage. Currently, the English Wikipedia has several 10,000's of articles containing the statement "*This article about ... is a stub. You can help Wikipedia by expanding it.*" For example, the famous database researcher Jennifer Widom has a 5-line page only. It does not know that she graduated from Cornell, worked for IBM Almaden Research, and made important contributions to semi-structured data models and stream query languages. However, all this information is easily available on the Web and could be used to expand the Wikipedia page. This problem, which applies to other knowledge bases as well, has recently led to the TREC challenge of *Knowledge Base Acceleration* (`trec-kba.org`): can we help authors/editors of knowledge bases to maintain encyclopedic contents in a timely manner, by giving intelligent recommendations about salient events and facts that should be considered for updating the knowledge-base contents about an entity.

1.2 Problem Statement

Our goal in this paper is to automatically compile such *salient contents* about entities, in order to ease knowledge bases maintenance. We refer to the output of this task as *"gems"*: text excerpts compiled from input Web sources. In order to avoid overwhelming users, our goal is to compile highly informative, concise gems about *long-tail or emerging entities.* A good set of gems should not contain contents unrelated to the entity of interest and should not have redundancies. Gems should usually be short (e.g., a few hundred words in total), but allow user-configurable length constraints.

For compiling gems, we consider as input not only well organized documents, but also news streams, postings in social media, speech-to-text transcriptions from online videos, etc. Such sources may have explicit structure in terms of sentences and paragraphs, but many do not have it and rather form *streams of words (or tokens)* without sentence boundaries. As a result, a gem may not have any sentence structure either. It may consist of concatenated incomplete sentences, image captions, list entries, headings, user comments, and others. However, the gem as a whole should be readable (by a smart human) in a self-contained manner. Table 1 shows a gem for the example entity Sutherland Falls (a lesser known waterfall in New Zealand).

The problem of computing entity-specific gems resembles issues in summarization, passage retrieval, document expansion, and novelty detection [3,4,22,23]. However, none of the prior work has tackled the combination where salient contents on an entity needs to be focused and novel yet must meet length bounds on the amount of returned information. Passage retrieval and document expansion are intermediate steps for user-oriented tasks (e.g., question answering or entity search); so their outputs are further processed by machines without space constraints. Summarization and novelty detection (e.g., for news streams), on the other hand, have focused on fixed granularities: sentences, paragraphs, or articles. Moreover, many prior methods critically rely on labeled training data. In contrast, our approach is unique in that it

- can tap into arbitrary word-stream sources including news feeds and speech-to-text transcriptions;
- does not use fixed granularities and can identify and compose arbitrary text units into entity's gems;
- is unsupervised and does not rely on any training data;
- directly supports the end-user task of knowledge maintenance and judiciously avoids overwhelming the user with too much information.

1.3 Approach and Contribution

In this paper we develop a full-fledged method for generating salient contents about a given entity, using minimal assumptions about the underlying sources. Our method, called *GEM* for Gem-based Entity-Knowledge Maintenance, identifies salient text pieces of variable granularity, using a budget-constrained optimization problem, which decides upon which sub-pieces of an input text should be selected for the final result. Each of these text pieces is an output gem.

GEM represents the input sources as a *stream of words (tokens)*, where each word is associated with a score for estimating how related the context of the word is to the entity of interest. In this way, text fragments that contain densely packed entity-related words will obtain a high score mass. For computing the per-word entity-relatedness scores, we use a short *seed text* about the entity. We assume that the seed text is provided by the user, but we expect it to be merely one or two sentences. Then, for each word in the text stream, we compute a language-model-based similarity between the entity seed text and the context of the word. To capture coherent text excerpts with high score mass while meeting the constraint that a user should not be overwhelmed with information, we develop a budget-constrained optimization algorithm mapped into an integer linear program. Our algorithm identifies variable-length fragments, which stand out by their saliency on the entity and their novelty with regard to the entity seed text.

Our approach can be applied to any real-world entity with a brief textual description to start from. For example, we could start with a long-tail entity briefly described in Wikipedia, or with a book covered in online communities like `librarything.com` or `shelfari.com`, with a singer's homepage on the Web, or with a text snippet about a newly emerging entity mentioned in news on recent events. Neither the seed texts nor the input sources are necessarily well-organized text. They may contain incomplete sentences, image captions, or social-media "slang". GEM processes all of these tokens uniformly, going beyond related work (e.g., on summarization) geared for sentences or paragraphs.

Oceania > New Zealand > South Island > Southland > Fiordland National Park > Sutherland Falls named for Donald Sutherland, a prospector who found the falls in 1880. William Quill, whom the lake feeding the falls was named for, is thought to be responsible for the first measurement of the falls which was attained by actually scaling the headwall next to the waterfall. New Zealand is a country which has a very high concentration of waterfalls. Unfortunately many of the best ones are isolated deep in the backcountry and are extremely difficult to access. We are fortunate then that the absolute best waterfall in the country is easily accessed via a very popular trail system ...

Table 1: An example gem for Sutherland Falls.

The fact that we do not pose any restrictions on the inputs, the entity seed and the sources, makes GEM highly versatile and widely applicable. For example, we can tap into speech inputs from video footage, by using standard methods for speech-to-text transcription (e.g., using software APIs by Microsoft or Google), and then running GEM on this text input. This text will be noisy because of transcription errors and will not contain any sentence or paragraph structure, yet GEM can handle it smoothly.

In summary, this paper makes the following novel contributions:

- formulating and modeling the new problem of compiling salient contents for a long-tail or emerging entity, to accelerate knowledge base maintenance;
- devising algorithms for extracting text gems from word-stream sources, by solving a budget-constrained optimization problem;
- conducting experiments that show the benefits of our GEM method in dealing with news streams and with query-based Web pages about long-tail entities;
- demonstrating a use-case with speech-to-text transcriptions as input.

2. COMPUTATIONAL MODEL

Given a short seed text about an entity of interest and input sources, our goal is to extract text excerpts from the sources, which are highly related to the input entity. We tackle this problem in the following steps:

- Step 1: Process the seed text of the entity, to build a statistical language model for the entity.
- Step 2: Represent the text sources as a stream of words, and for each word compute a relatedness score with regard to the entity (Section 3).
- Step 3: Run our method for extracting gems on the text stream, to obtain a set of text excerpts relevant for the entity (Section 4). If desired, further run novelty expansion (Section 5) or diversification (Section 6).

A seemingly obvious approach would be to first detect all mentions of the entity in the text stream, and then consider text windows around these mentions as gem candidates. However, this approach requires that the problem of named entity recognition and disambiguation is solved first, but this problem is very hard by itself, especially for long-tail and emerging entities. Our method does not require any entity detection in the underlying text.

3. RELATEDNESS FUNCTION

Assume we have an input entity and a text source. To select fragments from the text which are highly informative for the entity, we first estimate how related the text is to the entity at each position, by considering individual words (or tokens) as points of interest. Then, we select consecutive parts of the text that contain words highly related to the entity and have a high density of such words.

More formally, we represent the input text by its ordered sequence of words $S = (w_1, \ldots, w_n)$, with removed stopwords. Our goal is to estimate the *relatedness* to the entity e at each word $w_i, 1 \leq i \leq n$. However, individual words are meaningful only within their context: a window of surrounding words. So for each word w_i, we consider its *context* $c(w_i)$, which consists of a number of k words before w_i and k words after w_i, that is $c(w_i) = (w_{i-k}, w_{i-k+1}, \ldots, w_i, \ldots, w_{i+k-1}, w_{i+k})$.

We associate with each word w_i a *statistical language model* M_{w_i}. We estimate the parameters of M_{w_i} using the words in the context of w_i and their frequencies:

$$P(w|M_{w_i}) = \frac{\mathsf{tf}_{w,c(w_i)}}{\sum_{w' \in c(w_i)} \mathsf{tf}_{w',c(w_i)}}$$

We also build a statistical language model M_e for the entity e using its seed text in an analogical way. Finally, we compute a *relatedness function* which provides an estimate of how related the context of each word is to the entity:

$$f(w, e) = -KL(M_e||M_w)$$

Here $KL(M_e||M_w)$ is the Kullback-Leibler (KL) divergence between the language model of the entity (M_e) and the language model of the word w (M_w) given by:

$$KL(M_e||M_w) = \sum_t P(t|M_e) \log \frac{P(t|M_e)}{P(t|M_w)}$$

The use of KL divergence for retrieval and ranking of documents is state of the art in IR [26]. It usually measures the relatedness between a document and a query: low values of the KL divergence $KL(query|doc)$ denote high likelihood that the document generates the query and thus that it is informative/relevant for the query. In our setting the entity is in the role of a query and the context of a given word is used as a document. We apply Dirichlet smoothing using the entire Wikipedia corpus and $\mu = 500$.

To extract text gems for the entity, we use the relatedness scores of the words in the input text. Since high scores are assigned to words with context relevant to the entity, our goal is to select variable-length segments of the input text that contain many and densely packed words with high relatedness scores. To illustrate this idea, we show in Figure 1 an example text and the relatedness function computed over its word positions. We observe that certain parts of the text are more related to the entity than others.

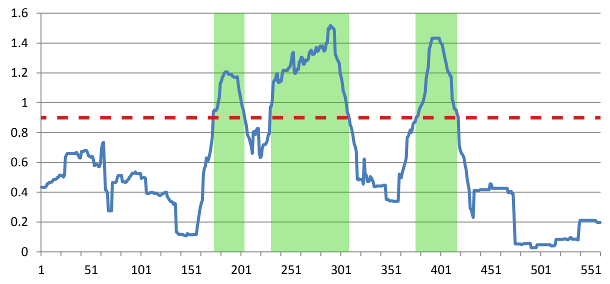

Figure 1: Text stream with its word positions on the x-axis and their relatedness values on the y-axis. Gems are extracted with a threshold value for relatedness and are shown in green.

4. EXTRACTION OF TEXT GEMS

Using the relatedness function we represent the input text as a sequence of word scores. We now explain how to extract gems with high word-score mass. We present two approaches, one based on thresholding on the score distribution over the word positions, and one based on an integer linear program with a specifically designed objective function.

4.1 Threshold-based Method

One option to select text gems is to use as an input a *user-specified threshold value*. Then we select text segments of words with relatedness values larger than the specified threshold (see Figure 1).

More formally, for a given threshold δ and input sequence of words $S = (w_1, w_2, \ldots, w_n)$, we select a gem $t = (w_i, w_{i+1}, \ldots, w_j), 1 \leq i \leq j \leq n$ as follows:

- $w_m \in t$ for all positions m such that $i \leq m \leq j$, and
- $f(w_m, e) \geq \delta$, for all $m, i \leq m \leq j$.

Note that for a given threshold value, there can be zero or more text segments extracted from a given text stream.

Gem Budget and Threshold Search. It is a difficult task for an ordinary user to choose suitable threshold values for different entities. The reason is that different entities can have relatedness functions with very different characteristics: smaller versus larger values, more peaks versus rather flat function, etc. To eliminate this complexity, we use as an input parameter a *user-specified budget of words* for the desired total number of words in the extracted gems. This parameter captures the idea that a knowledge-base editor or a user needs a bounded amount of highly informative text excerpts for extending or maintaining the knowledge about an entity; we use this parameter for the rest of paper.

The threshold-based algorithm searches for the smallest threshold such that the total number of the words in all extracted gems does not exceed the input budget of words. To this end, we observe that the word count in the selected text excerpts increases monotonically when the threshold decreases. Thus, we can run binary search on the threshold: in each step we first select the gems, following the above requirements, and then count their words. We define the search range by simply taking the minimum/maximum score over all words.

Gem Gaps and Minimum Length. Often two selected gems are within close proximity in the initial text. This means that the textual part in the gap between them is less related to the entity seed text, but the gap text may nevertheless contain new and interesting information about the entity. Recall that for computing the relatedness scores of the words, we use only the language model of the seed text. Thus, if the gap text talks about the entity but does not use its name and generally uses terminology that differs from that of the entity seed text, the relatedness score for the gap text is low. To capture this effect, we consider merging two neighboring gems together with their gap, but do this only when the selected gems are sufficiently close to each other in the input text. We consider two gems to be nearby if the distance between them is less than a certain number of words (e.g., the average length of one or two sentences, which varies between 10 and 60 words). In this case the selected gems are merged into a single one. To avoid very short gems, we require that all selected gems contain at least a certain number of words (e.g., the average length of a sentence).

We implement the above two heuristics by incorporating them into the word counting. Note that this does not violate monotonicity, and thus we can still use binary search.

4.2 ILP-based Method

We propose an alternative algorithm for extracting gems, using an integer linear program (ILP). Similarly to the threshold-based method, the input is a budget of words for the total gem length. However, instead of using heuristics like merging nearby gems, we model the task as an explicit optimization problem and develop a principled solution.

Our goal is to extract the most valuable set of gems T from the stream of words S, while observing the budget B. The ILP formulation needs to capture three requirements:

- 1) the accumulated per-word relatedness scores of the selected gems should be as high as possible (to select only highly informative gems),
- 2) longer gems are preferred over short ones (to select self-contained gems, and to merge nearby gems as the text between them may be relevant to the entity),
- 3) the total length of the gems in T does not exceed the budget B.

We introduce binary decision variables X_i, $i \in \{1, \ldots, n\}$, where n is the number of word positions in the stream S. $X_i = 1$, if the i-th word belongs to a selected gem, and $X_i = 0$ otherwise. Furthermore, we use binary variables $Y_{i,i+1}, i \in \{1, 2, \ldots, n-1\}$ for consecutive word pairs, which model the idea that we prefer (longer) sequences of words in the selected gems. $Y_{i,i+1} = 1$ if and only if $X_i = 1$ and $X_{i+1} = 1$. Although the $Y_{i,i+1}$ reflect only two adjacent words, by considering $Y_{i,i+1}$, $Y_{i+1,i+2}$, etc. together, we obtain the intended effect of rewarding the selection of longer sequences.

We can now precisely formulate the optimization problem by the following ILP model:

$$\begin{aligned} \text{maximize} \quad & \sum_i f(w_i, e) X_i + \alpha \sum_i Y_{i,i+1} \\ \text{subject to} \quad & \sum_i X_i \leq B \\ & Y_{i,i+1} \leq X_i \\ & Y_{i,i+1} \leq X_{i+1} \\ & Y_{i,i+1} \geq X_i + X_{i+1} - 1 \end{aligned}$$

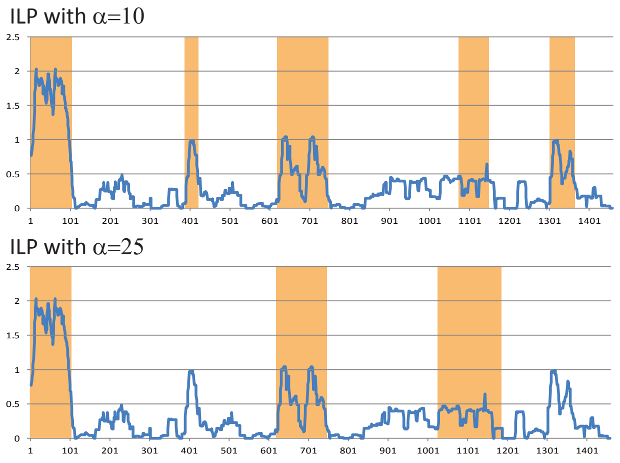

Figure 2: Comparison of selected gems using ILP with $\alpha = 10$ and with $\alpha = 25$ for budget of 400 words.

The first summand in the objective function aims to select gems which consist of words with high relatedness scores. The goal is to select only informative gems. The second summand rewards longer sequences of words. To explain this, consider two selected gems, with two words between them, which have low $f(.)$ scores, for example w_i and w_{i+1}. If we merge the two gems, then we set three more Y variables to 1: $Y_{i-1,i} = Y_{i,i+1} = Y_{i+1,i+2} = 1$. This means that the merged gem has larger objective score for a specific choice of the parameter α. The constraints that refer to both X_i and $Y_{i,i+1}$ encode that $Y_{i,i+1}$ is 1 if and only if both X_i and X_{i+1} are set to 1. Finally, $\sum_i X_i \leq B$ encodes that the total length of the gems should not exceed the budget B.

The parameter α controls the trade-off between the relevance of the gems and their length (see Section 5). We show experiments with different values of α in Section 7.

5. NOVELTY WITH REGARD TO SEED

One drawback of the methods from Section 4 is that they critically rely on the wording of the entity seed text. Since we use the language model of the seed text to compute the relatedness scores of the words, text fragments which do not have any words in common with the seed are always assigned very low scores. However, such text parts can still be relevant, especially when the seed text is very short. Furthermore, we would often miss out on novel information about an entity if expressed in terminology very different from the seed text, and capturing such novelty is exactly one of our key goals. Therefore, we present two extensions of our gem extraction methods to capture such novel information.

5.1 Large α in the ILP-based Method

The parameter α in our ILP method regulates the length of the gems and their relatedness to the entity seed. Low values of α reward highly informative text gems while large values of α reward longer gems. As discussed earlier, often highly informative gems contain only information which is already in the seed text. To capture more novel information about the entity, we increase the value of α.

We give an example for the influence of α on the extracted gems in Figure 2, and we show more experimental results in Section 7. In Figure 2 for $\alpha = 10$ our ILP algorithm selects

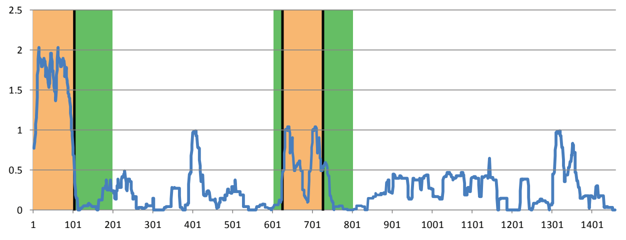

Figure 3: Example for gem expansion with $\alpha = 10$ for budget of 400 words. First, ILP with budget 200 words extracts the orange (light) gems, then the expansion appends the green (dark) parts.

short but highly relevant text fragments. On the same input text, using $\alpha = 25$ we retrieve fragments which are longer but still contain many related words. We notice, that the second and the fifth gems chosen for $\alpha = 10$ are not selected for $\alpha = 25$. Instead, the forth gem is expanded with more words. To decide which gems to ignore and which gems to expand, the ILP uses the relatedness scores of the words and their neighborhoods. For example, the first gem for $\alpha = 10$ is not expanded for $\alpha = 25$ as the words on its right have very low relatedness values.

5.2 Gem Expansion

An alternative approach for finding novel information, which works for the threshold-based method as well, is to expand each gem by appending its surrounding text.

Let h be a parameter that specifies how much we want to expand each gem. To obtain a final set of gems consisting of B words in total, we first extract gems with a budget of B/h words using one of the approaches from Section 4. Then we append to each gem its surrounding text so that its length grows h times. Note that the left and the right textual parts adjacent to the gem can relate differently to the entity. Thus, we first estimate their relevance to the entity. We initially take both textual parts $h - 1$ times larger than the gem length. We compute the relevance of each part as the negative KL divergence between the seed language model and the language model of its text. Finally, we choose the lengths of the left and the right parts proportionally to their relevance. We use $h = 2$, which we found to perform best after testing different values in the range [1.5,3]. Figure 3 shows an example using the presented expansion technique.

Note that longer gems are expanded with more words than shorter ones. Intuitively, long gems are very prominent for the entity, and thus the stream may contain more relevant information adjacent to them, without matching the terminology in the seed.

6. DIVERSIFICATION OF GEMS

By using the entity seed to extract gems, the algorithms from Section 4 select text fragments which are similar to the seed text. Naturally, this can lead to the extraction of gems which are highly similar among each other. In the extreme case, if the seed is very short and/or the Web does not provide much information about the entity, the extracted gems could be almost identical. Since our goal is to extract as much relevant information as possible, we need to extend our extraction methods for *diversification* of gems.

Our method is based on the Maximal Marginal Relevance approach (MMR) [4]. The MMR approach re-orders a set of documents, retrieved for a given query, by incrementally choosing the next document which has maximal *marginal relevance*, until a cardinality constraint is met. The marginal relevance of each document is a linear combination of its relevance and its dissimilarity with the already chosen documents. We follow this approach and adapt it to our setting.

Let T be a set of extracted gems for an entity e. To find a subset $S \subseteq T$ of gems which are (i) relevant to the entity, (ii) diverse among each other, and (iii) have a total length that does not exceed the budget B, we follow the steps:
1) Initialize S with the gem with highest relevance score;
2) Iterate over all gems in $T \setminus S$ and choose the gem t with maximal marginal relevance score:

$$t = argmax_{g \in T \setminus S}[\lambda rel(g,e) + (1-\lambda) min_{g' \in S} sim(g,g')]$$

3) Add the selected gem t to S;
4) If the gems in S contain less than B words, repeat steps 2, 3, and 4. Otherwise, remove the last words from t such that the total number of words in S is not more than B.

We compute the relevance of a given gem t by the negative KL divergence between the entity seed and the gem: $rel(t,e) = -KL(M_e||M_t)$, where M_e is the language model of the seed text and M_t is the language model of the gem.

To compute the similarity between two gems g and g', we use the square root of the Jensen-Shannon divergence between their language models M_g and $M_{g'}$: $sim(g,g') = \sqrt{JSD(M_g||M_{g'})}$, where $JSD(M_g||M_{g'}) = \frac{1}{2}KL(M_g||M) + \frac{1}{2}KL(M_{g'}||M)$, and $M = \frac{1}{2}(M_g + M_{g'})$. We apply this approach by first running some of the methods from Section 4 with larger budget, and then iteratively selecting gems, until we reach the desired budget.

7. EXPERIMENTS

We address two experimental scenarios: one with news articles and one with Web search. We compare GEM to its competitors which leverage paragraph and sentence boundaries in the input text. The methods under comparison are:

- the *GEM method*, configured in 4 different modes:
  - **ILP**: the ILP method from Section 4.2;
  - **ILP-EXP**: the ILP method from Section 4.2 with the gem expansion for novelty from Section 5.2;
  - **ILP-MMR**: the ILP method from Section 4.2 with diversification based on MMR (Section 6);
  - **THR-SEARCH**: the threshold-based method from Section 4.1;
- **PAR**: a method, which first extracts paragraphs using <p> tags, then ranks them by the negative KL divergence between the seed and the paragraph, and finally outputs the best paragraphs that fit into the budget;
- **PAR-MMR**: diversification of paragraphs with MMR;
- **SENT**: a method, which ranks sentences by the negative KL divergence between the seed and the sentence, and outputs the best sentences that fit into the budget;
- **SENT-MMR**: diversification of sentences with MMR.

The ILP implementation uses the Gurobi Optimizer (`www.gurobi.com`). To compute the relatedness function, for each word we consider its context, which consists of 10 words before the word and 10 words after it (i.e., $k = 10$).

Sir Ranulph Fiennes has begun his journey home after having to pull out of an expedition across Antarctica in winter because of frostbite.
Investigators have confirmed that the blast last week that ripped open a central Prague office building and injured some 40 people was caused by a gas leak.

Table 2: Examples of seed texts for events.

groups like al Shabaab in Somalia and Boko Haram in Nigeria. *Sir Ranulph Fiennes will be evacuated from the Antarctic after suffering severe frostbite, forcing him out of his latest expedition. The British explorer and his fellow adventurers were training to take part in the Coldest Journey, a six-month trek across the continent due to start next month. But the 68-year-old developed frostbite after he had an accident while skiing and had to use his bare hands to repair his ski bindings ...*

Table 3: Part of an extracted gem for an event. The relevant text is shown in *red*.

7.1 Experiments with News Articles

7.1.1 Experimental Setup

Data. We compiled a set of entities from `wikinews.org`, which consists of 30 emerging events from February to April 2013. 21 of these entities do not have Wikipedia articles. They include "Prague explosion injures dozens", "Ukraine plane crash landing kills five", and others. We consider such entities as long-tail entities. The remaining events are mentioned in existing Wikipedia articles (e.g., "Pierre Deligne is awarded with Abel prize", "British explorer Ranulph Fiennes leaves Antarctic expedition after frostbite", etc.).

Seed Text and Input Text. We compiled a set of articles by using wikinews articles and their "Sources" links. For each entity we labeled the articles that are relevant for it. For each entity we also chose one of its relevant articles, and used its first one to three sentences as a seed text for the entity. The input text for the compared methods consists of *all* collected articles, except for the articles used as seed texts. This way, the seed texts and the input text are disjoint. In total we have 50 news articles as an input text. The GEM methods use a single stream of words, by first shuffling the 50 news articles and then concatenating them. Table 2 shows examples for seed texts.

Quality Metrics. For each entity, we use the articles that we labeled as relevant for it. We measure which portions of these articles are extracted in the entity gems. We denote by R the text from all labeled relevant articles for a given entity, and by E the text from all gems extracted for this entity. From the word sequences of R and E we have removed all stopwords. For example, assume there is a single labeled relevant article for a given entity with word sequence $R = (w_1, w_2, \ldots, w_{10})$. Assume also, that there is a single extracted gem with text $E = (w_4, w_5, \ldots, w_{12})$. Then $|E \cap R| = 7$. We use the following metrics:

- **Text precision** measures the amount of extracted information (in terms of words) which is relevant to the entity: $\textit{text precision} = |E \cap R|/|E|$.
- **Text recall** measures the amount of relevant information (in terms of words) which is extracted: $\textit{text recall} = |E \cap R|/|R|$.

Method	Metric	Budget			
		100	200	300	400
ILP $\alpha = 20$	text precision	0.944	0.88	0.786	0.696
	text recall	0.367	0.612	0.739	0.82
	text F1	0.483	0.659	0.698	0.693
PAR	text precision	0.898	0.781	0.673	0.583
	text recall	0.322	0.496	0.614	0.679
	text F1	0.435	0.555	0.587	0.574
PAR-MMR	text precision	0.887	0.755	0.653	0.565
	text recall	0.314	0.487	0.602	0.669
	text F1	0.426	0.541	0.572	0.559

Table 4: Evaluation for news articles.

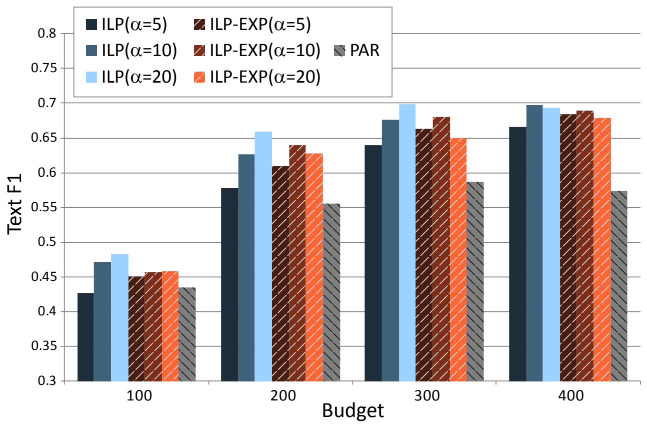

Figure 4: Evaluation for news articles of ILP and ILP-EXP with different values of α.

- **Text F1** is the harmonic mean of text precision and text recall:

$$\textit{text F1} = 2 \cdot \frac{\textit{text precision} \cdot \textit{text recall}}{\textit{text precision} + \textit{text recall}}$$

7.1.2 Results

Table 4 shows our experimental results comparing ILP with PAR and PAR-MMR for different sizes of the budget. ILP with $\alpha = 20$ is more effective than the paragraph-based methods by a large margin on all budget sizes. For ILP with $\alpha = 20$ and PAR, we performed two-sided paired t-tests w.r.t. F1 on all budget sizes: the p-values were <0.02. Table 3 shows an example gem computed by ILP with $\alpha = 20$.

Figure 4 shows the F1 measure for ILP and ILP-EXP for different values of the parameter α. Regarding ILP, increasing α leads to (1) selecting longer segments which contain more relevant information about the entity (improved recall), and (2) avoiding short segments which contain some words related to the entity, but have different topics otherwise (improved precision). Note that by improving the recall, we extract more novel information with respect to the entity seed. From Figure 4 we also notice that ILP-EXP is always more effective than the paragraph-based baseline.

Our experiments show that ILP-MMR slightly improves ILP only for small values of α. For a budget of B words, we first run ILP with budget $2B$, and then follow Section 6 to choose the best gems that fit into B. F1 for ILP-MMR with $\alpha = 5$ is 0.466, 0.623, 0.634, 0.675 for a budget of 100, 200, 300, 400 words, respectively.

Long-tail Entities	Standard Entities
Sutherland Falls Yumbilla Falls	Mahood Falls Hunlen Falls
Nicholas Pippenger David Eppstein	Frances Allen Samson Abramsky
“Lucky (memoir)” by A. Sebold “Rama II” by A. C. Clarke	“A Spot of Bother” by M. Haddon “Netherland” by J. O’Neill

Table 5: Examples of long-tail and standard entities.

Sutherland Falls: Sutherland Falls is a waterfall near Milford Sound in New Zealand’s South Island. At 580 meters (1,904 feet) the falls were long believed to be the tallest waterfall in New Zealand.
“The Glass Books of the Dream Eaters”: Gordon Dahlquist’s debut novel is a big, juicy, epic that will appeal to Diana Gabaldon fans (see her quote below) and lovers of literary fantasy, like Keith Donohue’s The Stolen Child. The Glass Books of the Dream Eaters begins with a “Dear Jane” letter in which Celeste Temple learns of the end of her engagement. Curiosity leads her to follow her fiance to London where she uncovers a secret.

Table 6: Examples of seed texts.

THR-SEARCH is consistently inferior than the ILP variants, with results similar to PAR. Varying the minimum segment size (in words) p and the maximum distance between merged gems q, we found out that larger values of q slightly improve the results. We obtained best results for $p = 10$ and $q = 60$. The F1 measure for THR-SEARCH is 0.458, 0.54, 0.586, 0.593 for a budget of 100, 200, 300, 400 words, respectively.

We observe that the news articles typically contain short paragraphs with a few sentences only. Since short paragraphs do not always contain enough characteristic information about the entity, the baselines do not achieve good results. The same observation holds for sentence-based extraction: the F1 measure for SENT is 0.361, 0.479, 0.511, 0.5 for a budget of 100, 200, 300, 400 words, respectively, and SENT-MMR returned worse results.

7.2 Query-based Experiments

In a second line of experiments, we used Google queries to obtain the input text for entities.

7.2.1 Experimental Setup

Data. We compiled three sets of test entities: 25 waterfalls, 25 computer scientists, and 25 books. We consider both long-tail entities and “standard” entities; see Table 5 for examples. The former are poorly covered in Wikipedia (usually marked as stub articles); the latter have below-average article lengths.

Seed Text and Input Text. For waterfalls and scientists, the seed text consists of the first one to three sentences from the respective Wikipedia article, ranging between 10 and 50 words in length. For books, the seed text was taken from the descriptions section of the respective `librarything.com` page, ranging between 50 and 120 words. Table 6 shows examples for seed texts. To gather input texts, we used the entity name (sometimes with a qualifier such as “Netherland Joseph O’Neill” instead of merely “Netherland”) for querying Google, fetched the top-10 results, extracted the text between their <p> tags, and concatenated it into a single stream of words. We excluded all pages from Wikipedia and `librarything.com`.

Method \ Budget	400	500	600
ILP	0.474	0.504	0.531
ILP-EXP	0.483	0.517	0.542
ILP-MMR	0.476	0.514	0.528
THR-SEARCH	0.449	0.487	0.519
PAR	0.457	0.491	0.518
PAR-MMR	0.462	0.49	0.516
SENT	0.432	0.471	0.501
SENT-MMR	0.458	0.489	0.517

Table 7: Evaluation in terms of phrase recall.

Ground Truth. For all test entities we compiled “ideally informative” texts as ground truth for quality metrics. For waterfalls and scientists, we used the rest of the respective Wikipedia article (excluding the seed text). For books, we used the respective Wikipedia article. This way, the ground-truth text is disjoint from the seed text for each entity. From these ground-truth texts we extracted noun phrases with the OpenNLP library (`opennlp.apache.org`), which we used as a gold standard for the quality of the gems. To ensure that phrases are truly informative, we filtered out all phrases that do not match any Wikipedia article title. For the 75 test entities, there are 3,330 noun phrases in total.

Quality Metrics. We use a recall-based metric computed over the relevant noun phrases for each entity. We measure the fraction of ground-truth phrases that are contained in the entity gems. Let R be the ground-truth text for an entity, and G the text in all computed gems for this entity. Let $F(D)$ be a binary vector representing phrases in a text D: $F^i(D) = 1$ if the i-th phrase is in D, and $F^i(D) = 0$ otherwise. Then,

$$\textit{phrase recall} = \frac{\langle F(R), F(G)\rangle}{\langle F(R), F(R)\rangle}$$

where $\langle ., .\rangle$ is the inner product of two vectors. Note that this metric is a special case of the n-gram based metric ROUGE-N [19] used in text summarization.

7.2.2 Results

Table 7 shows the experimental results for all compared methods, varying the budget of words. For all ILP variants, we show only the case with $\alpha = 20$, as this setting almost always performed best. THR-SEARCH was configured with minimum gem length of 10 words and a maximum distance between merged gems of 60 words, similarly to Section 7.1.

The main observation from Table 7 is that all our ILP variants outperform the baseline methods. Among the ILP methods, the differences are relatively small; ILP-EXP achieves the best results. To compare ILP-EXP and PAR, we performed two-sided paired t-tests which had p-values of 0.06, 0.06, and 0.07 for a budget of 400, 500, and 600 words, respectively. Table 1 shows an example gem computed by ILP with $\alpha = 20$ for budget of 400 words.

As for the inferior performance of the baselines, these methods suffer from their reliance on paragraphs or sentences. In the experiments, we often observed that relatively long paragraphs were selected, quickly exhausting the space budget and thus disregarding other valuable parts of the in-

Nicolas Checque (Web seed text): Nicolas Checque, the 28-year-old SEAL Team 6 member who was in the helicopter assault to free an American doctor from the Taliban, was killed by a single gunshot to the head.
Ravi Shankar (Wikipedia seed text): Ravi Shankar (7 April 1920 – 11 December 2012) often referred to by the title Pandit, was an Indian musician and composer who played the sitar, a plucked string instrument. He has been described as the best-known contemporary Indian musician.

Table 8: Examples for audio entities and their seeds.

put texts. On the other hand, our method did not consider well-formed and informative structures, which despite being rare in the dataset of Section 7.1, were more frequent here.

8. APPLICATION TO AUDIO STREAMS

We address the following use-case, demonstrating the benefits of GEM being independent of sentence or paragraph boundaries. Given an audio stream of news and an entity or topic of interest, retrieve excerpts of the audio stream that are related to this entity. For example, if users are interested in the latest news about the hurricane Sandy, they could be directly pointed to the specific parts of the audio stream where this topic is discussed.

To solve the problem, we utilize *speech transcriptions* provided by a standard speech recognition system. From the transcriptions, we extract relevant gems using the methods presented in this paper. Since speech transcription also returns the time positions of the transcribed words, we associate with each extracted gem its respective time interval in the input audio. This way, we retrieve audio segments relevant to a given entity.

8.1 Experimental Setup

Audio Streams. We collected 10 audio podcasts from NBC Nightly News for the days between December 9 and December 18, 2012. These audio streams provide reports of the most important international events that took place on the respective day. Each audio recording is approximately 20 minutes, some are longer.

Entities. To choose test entities, annotators listened to the complete podcasts and identified salient entities: events or people that are discussed. Our test data includes 10 entities: Nelson Mandela, Susan Rice, hurricane Sandy, Daniel Inouye, North Korea's satellite launch on December 12, gas explosion in West Virginia on December 11, plastic waste in Hawaii, Nicolas Checque, Ravi Shankar, and Jenni Rivera.

Seed Texts. For each test entity we compiled a seed text, which consists of the first 1 to 3 sentences from the respective Wikipedia page (bounding the total number of words to 50). For 4 entities we used seeds from manually retrieved Web pages, when there is no representative article for the person or event in Wikipedia (e.g., Nicolas Checque or the gas explosion in West Virginia on December 11). Examples for seed texts are shown in Table 8.

Relevant Audio Segments. While choosing the test entities from the audio podcasts, the annotators labeled time intervals, in terms of minutes and seconds, during which these entities are discussed. These timeframes are considered as ground-truth for the test entities.

The number of relevant timeframes varies across entities. Some topics are mentioned only once (e.g., the death of Ravi Shankar), while other topics are discussed several days (e.g., consequences from the hurricane Sandy). The relevant intervals can be short (1 or 2 minutes) or long (ca. 5 minutes).

Quality Metrics. We use the ground-truth timeframes for measuring gem quality. As each text fragment is associated with start and end time points, we compare the intervals of the extracted gems against the ground-truth intervals:

- **Time precision** measures the amount of extracted information (in terms of seconds) which is relevant to the entity. Let E be the set of extracted seconds, and R – the set of labeled relevant seconds. Then, $\textit{time precision} = |E \cap R|/|E|$
- **Time recall** measures the amount of relevant information (in seconds) which is extracted: $\textit{time recall} = |E \cap R|/|R|$
- **Time F1** is the harmonic mean of time precision and time recall:

$$\textit{time F1} = 2 \cdot \frac{\textit{time precision} \cdot \textit{time recall}}{\textit{time precision} + \textit{time recall}}$$

Transcriptions. To transcribe the news podcasts, we use the System.Speech namespaces in the Microsoft .NET Framework. As parameters we use the "en-US" culture and the grammar is "DictationGrammar". The result is a single stream of words; the speech recognizer does not provide any markup for sentences or paragraphs. We concatenated the transcriptions of all podcasts in our dataset into a single stream and fed it into our GEM method.

To emulate paragraph-based methods (for comparison), we used two modes for determining speech pauses. We varied the "SpeechRecognitionEngine.EndSilenceTimeout" property, which determines how long the speech recognizer waits until it finalizes the transcription and outputs a *recognized text segment.* Since the audio input is noisy and ambiguous, larger timeout (>1 sec) results in more robust result. The recognized segments are longer (40 to 60 words) with low variance. Smaller timeout (the default is 150 ms) results in text segments of highly varying lengths, ranging from a couple of words to 50 words.

Competitors. The methods under comparison are:

- **GEM**: our ILP approach from Section 4.2 (without novelty expansion or diversification);
- **RT-S**: recognizing text segments with short timeout for pauses;
- **RT-L**: recognizing text segments with long timeout for pauses.

The input to our GEM method is a single stream of words; the information about speech pauses is not used in GEM. To compute the relatedness function, for each word we consider 20 words before the word and 20 words after it (i.e., $k = 20$).

8.2 Results

Table 10 shows the experimental results. We configure the ILP method with $\alpha = 10$ ($\alpha = 20$ led to similar results). We observe that our method extracts gems with high time precision and achieves close to perfect time recall for larger budgets. GEM retrieves almost all relevant information about the entity, while avoiding to overload the user with addi-

Labeled Relevant Timeframe: (15:53 – 16:20)
Extracted Gem (15:22 – 16:26):
NBC news New York and again tonight with more on the web including the helpful resources the AARP you can find it all and NBC nightly news staff when we come back is a big complaint about television viewing finally getting taken care of an *ravi shankar has died first time most of this heard the sound of that music of the signs to the beatles and was george harrison first fell in love with the sound of the sitar thanks to shankar virtuoso player and composer people of the sound of the east and west really be composed the score to the film gandhi he died this week following heart surgery on longest surviving family a storm nora Jones ravi shankar was ninety two* was a big ball around here at the today show today you know the theory we all have a double out there somewhere on OS De

Table 9: Extracted segment from audio transcriptions using GEM for budget of 100 words. Start and end times are given in {minutes:seconds} format. The relevant text is shown in *red*.

tional information. In contrast, the two baseline methods retrieve segments which are significantly less related to the entity; even for large budgets their time recall is much lower than the results for GEM. This can be also noticed from the F1 measures of the compared methods. Table 9 shows an example of an extracted segment using our GEM method.

GEM captures the points in the audio stream when the entity is discussed and either expands these points or not, depending on the given budget. In contrast, since the baselines are limited to the pseudo-paragraphs given by the speech recognizer, they often select irrelevant contents. The reason is twofold: (1) the recognized text segments are very noisy as they come from speech transcription, and (2) their lengths are sometimes insufficient to judge if the text is related to the entity or not.

9. DISCUSSION

A major strength of our approach is that it does not pose any restrictions on the input sources. This makes it applicable to a wide variety of text inputs, such as social-media postings, speech-to-text transcriptions, conversational threads in online discussion forums, and more.

Our approach is also able to handle entities with ambiguous names. The entities are represented with their seed texts, which despite their brevity, turns out to provide enough information to uniquely identify entities among those with the same or similar names.

Regarding efficiency, our threshold-based method is computationally significantly cheaper than our ILP method. The former works in $O(n \log k)$, where n is the number of the words in the stream, and k is the desired precision, whereas integer programming is in general NP-hard. Thus, our ILP method does not scale up to large inputs. Research on efficiency and scalability issues is left for future work.

Our methods can be extended to consider the evolution of the input stream over time. To this end, we simply need to periodically re-compute entity gems over moving time windows of input text. We can also harness this time awareness to rank gems by recency.

Finally, note that in this work we focused on devising algorithms for extracting gems from *given* text sources. Orthogonal to this task is the retrieval of as many entity-related sources as possible. We did not address this issue here.

Method	Metric	Budget			
		100	200	300	400
GEM	time precision	0.765	0.77	0.667	0.558
	time recall	0.439	0.755	0.885	0.94
	time F1	0.494	0.692	0.698	0.647
RT-S	time precision	0.663	0.488	0.411	0.341
	time recall	0.407	0.542	0.632	0.674
	time F1	0.445	0.455	0.448	0.41
RT-L	time precision	0.765	0.614	0.488	0.396
	time recall	0.497	0.675	0.762	0.784
	time F1	0.541	0.58	0.543	0.483

Table 10: Evaluation for audio streams.

10. RELATED WORK

Content Enrichment. The expansion of a document collection in [23] is used to enhance the quality of a question answering system. The authors extract a set of candidate paragraphs based on HTML markup, and rank them using Logistic Regression. The expanded corpus is not intended for humans and is not bounded. In contrast, we aim to retrieve a bounded amount of highly informative related information. Moreover, we consider (1) no markup-dependence on the input text, (2) novelty w.r.t the seed text, (3) diversification, and (4) independence of training data.

An algorithm for improving information retrieval of "short texts" through aggressive document expansion is proposed in [10]. Each short text is submitted as a pseudo-query in a large corpus, and the obtained results are used to enhance the language model of the initial document.

The goal in [2, 20] is to extract key concepts from a given text, which are then linked to external sources such as Wikipedia. In both works, the key concepts are in the form of short keywords and keyphrases. [15] automatically enriches Wikipedia facts with supporting external links, comparable to those manually selected by Wikipedia authors.

Passage Retrieval. The goal of passage retrieval is to extract passages that are relevant for a given query. There are mainly two approaches. First, during passage retrieval there is no knowledge about the query: [3, 23]. Passages are extracted based on sentences, paragraphs, HTML tags, n-grams, etc. Only then, the retrieved passages are evaluated using standard retrieval methods (e.g., language models) for their relevance to the query. Second, during passage retrieval there is prior knowledge about the query: [6, 14, 18]. In [6, 14] the passage locations are fixed after the query is evaluated, such that they have highest relevance to the query. Instead of using predefined passages, the authors analyze the shortest segments (covers) in the text which contain all query words. However, typically the used queries consist of a few keywords only, while in our work the seed text can be of arbitrary length. In [18] passages are extracted by first assigning to each word a probability score, which depends on the query, and then selecting sequences of words with high scores. The probability scores of the words depend only on the words themselves; the surrounding words are not considered. In our work, we use as an input the entity seed and we do not consider predefined passages.

Text segmentation is the task of retrieving parts of the input text, which are semantically coherent. Existing approaches divide the given text when there is a shift from one topic to another by using change in the vocabulary [5,24] or statistical topic analysis [16,21].

Text Summarization. Prior work on summarization [22], and especially extractive summarization, is naturally related to our setting. However, the fact that summaries are intended for human readers mandates that summaries consist of entire sentences. This is a fundamental difference to our knowledge-oriented setting where any text snippet (e.g., captions) and even semi-structured fragments (e.g., table rows) can contribute to valuable gems. Nevertheless, the specific directions of multi-document summarization [12,13,25], where diversity matters, and query-driven summarization [7,8,17], where thematic focus matters, are applicable to the problem of gathering content gems. In our experimental studies, we capture the essence of many methods along these lines by extracting, ranking, and diversifying sentences from the input documents.

Diversification. Search results diversification [1,4,9,11] has been an established problem in ranking algorithms for Web search. One early work in this direction is the "Maximal Marginal Relevance" approach [4], which we exploit in Section 6. [1] proposes a diversification objective which tradeoffs relevance and diversity. The approach aims at minimizing the risk of dissatisfaction of the user, given that there exists a categorical information about the queries and the documents. In contrast to this work, we are not given with such categorical information. [11] develops an axiomatic approach to characterize different diversification functions.

11. CONCLUSION

The work in this paper is a contribution to aid knowledge communities in timely and convenient maintenance of knowledge about entities. Prior work related to this task assumes that input documents are well-formed text with sentence and paragraph markup. Our GEM method does away with this limitation and provides a suite of techniques that can cope with arbitrary input streams of tokens, including news streams or audio transcriptions from videos. The experimental results presented in this paper demonstrate the viability of this novel approach. We believe that with the ongoing deluge of multimodal contents on the Web, in social media, and in enterprises, such departures from established paradigms are vital to cope with the ever-increasing pace of producing new information and knowledge.

12. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM '09*, pages 5–14.

[2] R. Agrawal, S. Gollapudi, A. Kannan, and K. Kenthapadi. Data mining for improving textbooks. *SIGKDD Explor. Newsl.*, 13(2):7–19, 2012.

[3] J. P. Callan. Passage-level evidence in document retrieval. In *SIGIR*, pages 302–310, 1994.

[4] J. Carbonell et al. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[5] F. Y. Y. Choi. Advances in domain independent linear text segmentation. In *NAACL*, pages 26–33, 2000.

[6] C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *SIGIR*, pages 358–365, 2001.

[7] J. M. Conroy, J. D. Schlesinger, and D. P. O'Leary. Topic-focused multi-document summarization using an approximate oracle score. In *COLING-ACL*, pages 152–159, 2006.

[8] H. Daumé, III and D. Marcu. Bayesian query-focused summarization. In *ACL*, pages 305–312, 2006.

[9] M. Drosou and E. Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, 2010.

[10] M. Efron, P. Organisciak, and K. Fenlon. Improving retrieval of short texts through document expansion. In *SIGIR*, pages 911–920, 2012.

[11] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW '09*, pages 381–390.

[12] A. Haghighi and L. Vanderwende. Exploring content models for multi-document summarization. In *HLT-NAACL*, pages 362–370, 2009.

[13] S. Harabagiu and F. Lacatusu. Using topic themes for multi-document summarization. *ACM Trans. Inf. Syst.*, 28(3):13:1–13:47, 2010.

[14] M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *J. Am. Soc. Inf. Sci. Technol.*, 52(4):344–364, 2001.

[15] C. W. Leong and S. Cucerzan. Supporting factual statements with evidence from the web. In *CIKM*, pages 1153–1162, 2012.

[16] H. Li and K. Yamanishi. Topic analysis using a finite mixture model. *Inf. Process. Manage.*, 39(4):521–541, 2003.

[17] P. Li, J. Jiang, and Y. Wang. Generating templates of entity summaries with an entity-aspect model and pattern mining. In *ACL*, pages 640–649, 2010.

[18] Q. Li, K. S. Candan, and Y. Qi. Extracting relevant snippets fromweb documents through language model based text segmentation. In *WI*, pages 287–290, 2007.

[19] C.-Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*, pages 71–78, 2003.

[20] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242, 2007.

[21] H. Misra, F. Yvon, J. M. Jose, and O. Cappe. Text segmentation via topic modeling: an analytical study. In *CIKM*, pages 1553–1556, 2009.

[22] A. Nenkova and K. McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233, 2011.

[23] N. Schlaefer, J. Chu-Carroll, E. Nyberg, J. Fan, W. Zadrozny, and D. Ferrucci. Statistical source expansion for question answering. In *CIKM*, pages 345–354, 2011.

[24] M. Utiyama and H. Isahara. A statistical model for domain-independent text segmentation. In *ACL*, pages 499–506, 2001.

[25] X. Wan and J. Yang. Multi-document summarization using cluster-based link analysis. In *SIGIR*, pages 299–306, 2008.

[26] C. Zhai. *Statistical Language Models for Information Retrieval.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2008.