Visual Analytics for Topic Model Optimization based on User-Steerable Speculative Execution

Supplementary Material: Speculative Execution

Mennatallah El-Assady[1,2], Fabian Sperrle[1], Oliver Deussen[1], Daniel Keim[1], and Christopher Collins[2]

[1]University of Konstanz, Germany
[2]University of Ontario Institute of Technology, Canada

In order to get a better understanding of the concrete speculative execution implementation in our paper, we describe different factors in more detail. First, section 1 describes different quality measures, most of which have been implemented in the VisArgue framework. Section section 2 details speculative execution, with subsection 2.1 focusing on the available optimization strategies and subsection 2.2 on the model quality monitor and its automated triggers.

1 Topic Model Quality Measures

A diverse set of quality metrics enables us to capture different notions of topic model quality. In the following, we will give an overview of different metrics available in the literature as well as present some new metrics. A lot of the presented metrics have been implemented in the VisArgue framework, as far as this was possible or useful with our focus on non-probabilistic incremental topic models. Some metrics rely on meta information of probabilistic models—their implementation has been postponed for now. To accurately describe the metrics and their formulas define a few symbols and functions:

$T = \{t_1, t_2, \ldots, t_n\}$	The list of n available topics
$t = \{d_1, \ldots, d_k\}$	A topic containing k documents
$C = \{d_1, \ldots, d_i\}$	The corpus containing i documents
$V = \{w \mid w \in \bigcup_{d \in C}\}$	The corpus vocabulary
$V^{(t)} = \left(v_1^{(t)}, \ldots, v_M^{(t)}\right)$	The topic descriptor of t

The topic descriptor contains the M most probable keywords in the topic, in decreasing order of frequency. In addition to the previous symbols we define:

$$\begin{aligned}
D(v) &= ||\{d \mid v \in d\}|| \\
D_t(v) &= ||\{d \mid v \in d \wedge d \in t\}|| \\
D(v,w) &= ||\{d \mid v \in d \wedge w \in d\}|| \\
D_t(v,w) &= ||\{d \mid v \in d \wedge w \in d \wedge d \in t\}||
\end{aligned}$$

Name	Description
Coherence	Measures how well common keywords in a topic descriptor predict less common ones by determining the conditional co-occurrence probabilities.
Separation	Measures how well the descriptors of a topic describe the corpus without the respective topic. If they do, the topic is too general.
Distinctiveness	The ratio between separation and coherence. Measures that documents are similar to those in the same topic but different from all others.
Pointwise Mutual Information	Measures the probability of co-occurrence of the most frequent keywords of the documents of a topic.
Certainty	Verifies that the most frequent keywords of the documents of a topic are unique to their topic, and do not appear in the descriptors of other topics.
Variance	Measures to which degree the documents of a topic are equally similar to all other documents in the topic (i.e. whether a cluster is uniform).
Topic Branching	The average number of subtopics for each topic. Measures how general (low topic branching) or specific (high topic branching) subtopics are.
Node Branching	The average number of additional subtopics at any tree level. Very low values indicate document chaining.
Compactness	The ratio between leaf nodes and inner nodes of the tree measures how many additional topic hierarchies have been introduced into the model. More compact trees are not as deep.
Speaker Count	The average number of unique speakers/authors in a topic.
Topic Size	The average number of documents in a topic.
Topic Count	The number of topics in the model.

Table 1: Overview of the quality metrics that have been implemented in the VisArgue framework. A further six metrics have been reviewed but are not currently implemented. The respective reasons against an implementation are giving in the following section.

$D(v)$ is the number of documents in the corpus containing the word form v, and $D(v,w)$ the number of documents containing both word forms of v and w. Analogously, $D_t(v)$ is the number of documents in the topic t containing the word form v, and $D_t(v,w)$ the number of documents in t containing both word forms of v and w.

While the previous notations were algorithm agnostic, we also need the following symbols that are specific to the ITHM algorithm and its implementation:

$$\begin{aligned}
k(t)&: \text{ The root node of the sub tree defined by topic t}\\
n(t)&: \text{ All tree nodes of a topic t}\\
i(t)&: \text{ Inner tree nodes of topic t}\\
f(x)&: \text{ Direct (non-transitive) child nodes of node x}\\
l(t)&: \text{ Leaf tree nodes of topic t. } ||t|| = ||l(t)||
\end{aligned}$$

and these that are useful for probabilistic models:

$$\begin{aligned}
N(w,d,t)&: \text{ The frequency of } w \text{ in } d \text{ assigned to } t\\
N(d,t) &= \sum_{w \in d} N(w,d,t)
\end{aligned}$$

Probabilistic models typically assign topics t to keywords w. The topic of a document d is then determined by its keyword distribution. However, keywords are not necessarily unique to a single topic, leading to the third parameter t in $N(w,d,t)$. $N(d,t)$ is thus the sum of frequencies of keywords belonging to topic t in document d.

Whenever the *cosine similarity* between two topics or documents is calculated, we refer to the *cosine similarity* between their respective term frequency vectors. Note that the dimensionality and order of these term frequency vectors are shared between topics and documents. This makes it possible to quickly calculate the similarities between a document and an entire topic, as well.

Document Entropy First, the probability of a document given a topic is defined as follows:

$$P(d \mid t) = \frac{N(d,t)}{\sum_{d' \in C} N(d',t)}$$

This is the sum of keyword frequencies of keywords in the document belonging to topic t, normalised by their frequencies in all documents in the corpus. The document entropy is the entropy of the probability distribution of documents in a topic [9].

$$\text{document\_entropy}(t) = H(P(d \mid t))$$

A topic with low document entropy is very specific, and likely contains only a few documents. In contrast, a topic with high entropy is more spread out across the corpus. However, the *mallet* authors reason that low entropy for a topic is not necessarily good, but could also be an effect of incorrectly cleaned input data containing few documents in different languages or metafiles. This metric is not applicable to non-probabilistic topic models where each document can only be in at most one topic at any time.

Word Length *Mallet* also defines the topic descriptor keyword length as a metric [7]:

$$\text{word\_length}(t) = \frac{\sum_{v \in V^{(t)}} \text{len(v)}}{M}$$

They argue that the word length is a good proxy for document specificity, as more useful descriptor keywords tend to be longer. In our opinion this metric is questionable—the selection of keywords is in the algorithms responsibility, which can easily select long keywords even for bad topics. Additionally, it is easy to imagine short names and acronyms holding relevant information and being valuable as topic keywords. As a consequence of our reservations, this metric is currently not implemented in VisArgue.

Uniform Distance This metric measures the specificity of a topic as the distance between the topic's word distribution and a uniform distribution, and has been taken from *Mallet* as well [7]:

$$\text{uniform\_dist}(t) = \sum_{w \in t} P(w \mid t) \cdot \log \frac{P(w \mid t)}{\frac{1}{||V||}}$$

The distance is calculated as the Kullback-Leibler Divergence between the topic's word distribution and the uniform distribution. Larger divergence values show a more significant difference between the two distributions and indicate a more specific topic. However, particular attention is necessary for very skewed probability distributions that will always return small divergence values.

Corpus Distance Instead of calculating the distance to the uniform distribution, to get the corpus distance we calculate the distance to the word distribution over the entire corpus, that is all topics [7]:

$$\text{corpus\_dist}(t) = \sum_{w \in t} P(w \mid t) \cdot \log \frac{P(w \mid t)}{P(w \mid V)}$$

Again, larger divergence values mean that the topic is more distinct, while smaller ones indicate that it is closer to a random sample from the corpus. As the *mallet* authors note, this metric is strongly correlated with topic size, as larger topics are more likely to contain more words from the corpus vocabulary. As such, it is not suited for use in our streaming system where topics are naturally getting bigger over time. The metric would just record lower and lower values over time, without giving any indication of the models actual quality. Consequently, this metric has not yet been implemented either.

Word Mover's Distance A common problem with distances in ”bag of word” models is that two documents can be semantically close while having no shared terms. Take, for example, the two sentences ”Donald Trump is talking to the media” and ”The president is holding a press conference”. Despite being similar, they have a very large or even infinite distance in bag-of-word distance metrics. Kusner *et al.* thus propose the ”Word Mover's Distance” (WMD) [6]. It adapts the well known Earth Mover's Distance and incorporates *word2vec* to identify similar terms.

Assuming a *word2vec* embedding matrix $X = \mathbb{R}^{d \times n}$ for a vocabulary of n words, let $x_i \in \mathbb{R}^d$ the embedding of word i in a d-dimensional space. The distance between two words i and j is then defined as their euclidian distance in the d-dimensional space:

$$c(i,j) = ||x_i - x_j||_2$$

Documents are represented as normalized bag of word vectors $d \in \mathbb{R}^n$, with d_i the normalized frequency of word i in document d. The authors then define a flow matrix $T \in \mathbb{R}^{n \times n}$ that saves in $T_{i,j}$ how much of word i ”moves” to word j between two documents d and d'. One word can move to a single word as a whole, or partially to different words.

The Word Mover's Distance is then computed as

$$\text{wmd} = \min_{T \geq 0} \sum_{i=1}^{n} \sum_{j=1}^{n} T_{i,j} c(i,j)$$

with the two constraints

$$\begin{aligned} \sum_{j=1}^{n} T_{i,j} &= d_i \qquad \forall i \in \{1, \dots n\} \quad \text{and} \\ \sum_{i=1}^{n} T_{i,j} &= d'_j \qquad \forall j \in \{1, \dots n\}. \end{aligned}$$

These two constraints ensure that the probability mass of all words is moved completely from the first document and that the incoming probability mass for each keyword in the new document is equal to its normalised frequency.

The authors claim that WMD significantly outperforms many other document similarity metrics due to the high quality of *word2vec* embeddings. This seems logical, as simpler bag-of-word models lack the additional information encoded in the *word2vec* model.

In 2016, Huang et al. improved the performance of WMD even further by introducing a supervised learning component in the S-WMD metric [5]. Using labelled training data, they automatically learn individual keyword weights as well as a transformation of the *word2vec* space that brings keywords from documents with the same labels closer together. Using S-MVD, they are—when averaged over different test corpora—able to outperform any other currently existing document distance metric.

Topic Word Distribution Size Probabilistic topic models often keep a matrix $\beta \in \mathbb{R}^{n \times k}$ containing the probability $p(\text{word} \mid \text{topic})$ for all n words and k topics. In order to determine the “topic words” of a given topic we evaluate

$$\begin{aligned} \text{tw}(t_k) &= \{w \mid \text{argmax}_i \beta_{w,i} = k\} \\ \text{topic\_size}(t) &= ||\text{tw}(t)|| \end{aligned}$$

We first assign all words to their most probable topic and then count the number of words in each topic's set. The result is a metric that shows how general a topic is, with larger values indicating more general topics. Possible values for a topic can range from zero to the size of the vocabulary if one topic is the most likely for all keywords. As this metric only makes sense on probabilistic models, it has not been implemented in the scope of this paper.

Coherence Already in 2011 Mimno *et al.* introduced their version of topic coherence, often referred to as *UMass Coherence* based on the authors' research positions at the University of Massachusetts [8]. UMass Topic Coherence is very similar to pointwise mutual information and only differs in one term:

$$\text{coherence}(t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \left(\frac{D\left(v_m^{(t)}, v_l^{(t)}\right) + \varepsilon}{D\left(v_m^{(t)}\right)} \right)$$

To measure the quality of the model we calculate the weighted average of the coherence of all topics:

$$\text{coherence}(T) = \frac{\sum_{t \in T} \text{coherence}(t) \cdot ||t||}{\sum_{t \in T} ||t||}$$

This formula is, however, aimed at probabilistic topic models where documents can appear in multiple topics. When using IHTM, we always assign a document to at most one topic and thus modify the formula to

$$\text{coherence}(t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \left(\frac{D_t\left(v_m^{(t)}, v_l^{(t)}\right) + \varepsilon}{D_t\left(v_m^{(t)}\right)} \right)$$

and only count word occurrence and co-occurrence frequencies for those documents belonging to the current topic, instead of the entire corpus.

While pointwise mutual information (see below) measures the ratio between the co-occurrence frequencies normalised by the product of the two individual document frequencies, coherence omits the second normalising term and only normalises the co-occurrence frequency with the document frequency of the first of the two terms each time. The authors claim that this lead to better performance in their expert annotation study. The result is equivalent to the conditional probability of the less frequent word, given the more frequent one. To avoid taking the logarithm of 0 the parameter ε has been added. In the original implementation by Mimno *et al.* it is set to 1. However, Stevens *et al.* ran an extensive study comparing different topic modelling algorithms and different implementations of coherence [11]. They conclude that a smaller ε is favourable as this puts more emphasis on words that do not co-occur within the topic. As a consequence we follow the popular *Mallet* library and set ε to 0.01.

One issue of coherence is that it works well for very small topics with only a few, very similar documents. The fewer documents a topic contains, the higher the chances of their keywords co-occurring, at least for similar documents. On the other side of the spectrum, large and general topics tend to achieve good coherence values as well, as they contain generic keywords that appear in other documents with higher frequency. To mitigate this issue, Nikolenko *et al.* propose "tf-idf-coherence" [10]:

$$\text{c}_{\text{nikolenko}}(t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{\sum\limits_{d \in t} \text{tf-idf}\left(v_m^{(t)}, d\right) \text{tf-idf}\left(v_l^{(t)}, d\right) + \varepsilon}{\sum_{d \in t} \text{tf-idf}\left(v_m^{(t)}, d\right)}$$

We chose not to implement this approach. VisArgue already provides word frequencies based on a user-selected function, such as tf-idf or log likelihood ratio. While this scoring will not directly influence the coherence values, it influences the insertion of documents into the model as well as the selection of the keywords for the topic's descriptor vector.

Separation As outlined above, topics can score well on the coherence metric without being perceived as coherent by the users. Additionally, excellent coherence scores mean very little if the entire corpus is very similar. For an extreme example imagine the whole corpus consisting of repetitions of the same document. Now, it does not matter which arbitrary topic structure is created by the algorithm. All topics will have a perfect coherence score, while it is easy for users to spot that the documents should have been in the same topic (or removed during preprocessing, but that is not the point of this example). We propose a new metric called separation and derive it from the coherence calculation. It measures how well the descriptor vector of a given topic describes the rest of the corpus without the current topic.

First, we introduce two shorthand notations to avoid bloating the following formulas unnecessarily:

$$D_{T\setminus t}(v,w) = D(v,w) - D_t(v,w)$$
$$D_{T\setminus t}(v) = D(v) - D_t(v)$$

The separation of a topic is then defined as follows:

$$\text{separation}(t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \left(\frac{D_{T\setminus t}\left(v_m^{(t)}, v_l^{(t)}\right) + \varepsilon}{D_{T\setminus t}\left(v_m^{(t)}\right) + \varepsilon} \right)$$

We calculate the weighted average between topics to measure the quality of the entire model:

$$\text{separation}(T) = \frac{\sum_{t\in T} \text{separation}(t) \cdot ||t||}{\sum_{t\in T} ||t||}$$

We add a parameter ε both in the nominator and denominator to avoid both taking the logarithm of zero, and dividing by zero. The idea is that those general topics that are not separated from the rest of the corpus well have generic topic descriptors, which in turn describe the remaining corpus reasonably well. Analogous to coherence, separation values are always negative. Larger values occur when the topic's descriptors do not appear together outside of the topic and show a better separation, while smaller ones indicate that the topic in question was too generic.

Distinctiveness As motivated in the introduction to coherence and separation above, we want to get distinct, that is coherent but separate, topics. Consequently, we define distinctiveness as the ratio between separation and coherence:

$$\text{distinctiveness}(T) = \frac{\text{separation}(T)}{\text{coherence}(T)}$$

Both separation and coherence values are logarithms of probabilities and are always negative, thus making our distinctiveness metric always positive. Because separation is optimised towards large negative values, and coherence towards zero the resulting distinctiveness value increases as any of the two metrics improve.

Pointwise Mutual Information Despite Mimno *et al.* claiming better results with coherence, we still wanted to include PMI in the list of available measures to verify their findings on our test data. Our tests have confirmed PMI to be less reliable and more erratic than coherence. As expected, PMI is strongly correlated with coherence but has a higher degree of change between two points in time than coherence. Originally, PMI is defined as follows for a single topic:

$$\text{pmi}(t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(v_m^{(t)}) \cdot D(v_l^{(t)})}$$

and for the model:

$$\text{pmi}(T) = \frac{\sum_{t\in T} \text{pmi}(t) \cdot ||t||}{\sum_{t\in T} ||t||}$$

Again, like with coherence, we modify the formula to only count co-occurrence frequencies within the documents belonging to the topic, as they cannot be contained in multiple topics.

$$\text{pmi}(t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D_t(v_m^{(t)}, v_l^{(t)}) + 1}{D_t(v_m^{(t)}) \cdot D_t(v_l^{(t)})}$$

Bouma *et al.* recommend this change and claim that calculating the document counts for keywords just over the current topic instead of the entire corpus led to better results in their tests [1].

Variance This metric measures the variance of the pairwise similarities between all topics [3]. We first calculate the similarities:

$$\bar{s} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{i-1} \text{cosine\_similarity}(t_i, t_j)}{\frac{n(n-1)}{2}}$$

and then their variance:

$$\text{variance}(T) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{i-1} (\text{cosine\_similarity}(t_i, t_j) - \bar{s})^2}{\frac{n(n-1)}{2}}$$

The idea is that we try to achieve equidistant topics to avoid having multiple similar topics that could be combined into a single one. This metric thus also measures a degree of distinctness and separation between the individual topics.

Topic Certainty In addition to the topic descriptors defined above, we define document descriptors $V(d)$ in order to calculate the topic certainty introduced in our previous work [4]. To do so, we first collect all word forms in a document that are part of a topic descriptor of any topic:

$$H(d) = \left\{ w \,\middle|\, w \in d \wedge w \in \bigcup_{t=1}^{n} V^{(t)} \right\}$$

The ordered document descriptor is then defined as follows:

$$V^{(d)} = \left(v_1^{(d)}, \ldots, v_M^{(d)} \,\middle|\, v_k^{(d)} \in H(d) \wedge c(v_i) \leq c(v_j) \iff i \leq j \right)$$

Here, $c(v)$ is the term frequency of keyword v in the corpus. For each document, we select all keywords that appear in at least one topic descriptor, order them with decreasing term frequency just as the topic descriptors, and save them as the document descriptor vectors.

For each topic, we calculate the average distance between the topic descriptor and all document descriptors of documents in the topic. The used distance function *rwdp* will be presented below.

$$\text{avg\_rwpd'}(t) = \frac{\sum_{d \in t} \text{rwpd}(V^{(d)}, V^{(t)})}{||t||}$$

The resulting distances are normalised to $[0, 1]$:

$$\text{avg\_rwpd}(t) = \frac{\text{avg\_rwpd'} - \min_{k \in T}(\text{avg\_rwpd'}(k))}{\max_{k \in T}(\text{avg\_rwpd'}(k)) - \min_{k \in T}(\text{avg\_rwpd'}(k))}$$

Again, we use a weighted average of the scores of all topics to measure model quality:

$$\text{certainty}(T) = 1 - \frac{\sum_{t \in T} \text{avg\_rwpd}(t) \cdot ||t||}{\sum_{t \in T} ||t||}$$

As a distance function in the above calculations the “ranked and weighted penalty distance” (rwpd) is used [3]. It was introduced by El-Assady in 2015. First, two helper functions for the relative rank and weight of keywords are defined:

$$d(i,j) = \frac{|i-j|}{M}$$
$$w(M,k) = \sqrt{M} - \sqrt{k}$$

$d(i,j)$ is the normalised distance between two index positions i and j in a vector of length M. $w(M,k)$ returns an importance-weight for the k-th keyword in the vector of length M.

The penalty is then computed like this:

$$p(V_i^{(d)}, V_j^{(t)}) = \begin{cases} d(i,j) \cdot (w(M,i) + w(M,j)) \cdot p & \text{iff } V_i^{(d)} = V_j^{(t)} \\ w(M,i) \cdot p & \text{iff } V_i(d) \notin V^{(t)} \\ w(M,j) \cdot p & \text{iff } V_j(t) \notin V^{(d)} \end{cases}$$

The “penalty factor” p has been set to 1 in both the original implementation and our framework. If a keyword from the document descriptor is not contained in the topic descriptor, or vice-versa, the weight associated with the index position of the wrong keyword is taken as the penalty. If a keyword is contained in both vectors at different index positions, the rank difference is the penalty.

The ranked and weighted penalty distance between two vectors is then defined as follows:

$$\text{rwdp}(V^{(d)}, V^{(t)}) = \sum_{i=1}^{M} \sum_{j=1}^{M} p\left(V_i^{(d)}, V_j^{(t)}\right)$$

This metric measures how distinct different topics are, by asserting that their keywords are not interchangeable and do not appear in all documents at the same time, preventing very general topics without added information.

Tree Compactness We define tree compactness as follows:

$$\text{compactness}(t) = \frac{||t||}{||i(t)||}$$
$$\text{compactness}(T) = \sum_{t \in T} \frac{||t||}{||i(t)||}$$

Tree compactness is the ratio between (non-backbone) leave nodes (i.e. documents) and inner nodes for each topic. It captures the number of intermediate hierarchy levels that have been introduced. Intuitively, we would like to see compact trees, with the necessary amount of hierarchy, but without superfluous hierarchies or even node chains being added by the algorithm. The particular case of topic chains, as well as an appropriate optimisation strategy, will be elucidated in subsection 2.1.

Topic Branching Factor For all topics, this metric measures the average number of direct child nodes.

$$\text{topic\_branching} = \frac{\sum_{t \in T} ||f(k(t))||}{n}$$

It is equivalent to the “fan out” of topic subtree root nodes, and thus strongly correlated with the overall tree compactness.

Node Branching Factor The node branching factor is similar to the topic branching factor but is calculated as the average over all inner nodes in the tree, not just topic nodes.

$$\text{node\_branching} = \frac{\sum_{t \in T} \sum_{m \in i(t)} ||f(m)||}{\sum_{t \in T} (||i(t)||)}$$

It is thus the average fan out of all (inner) nodes of the tree.

Topic Size The average number of documents in all topics.

$$\text{topic\_size} = \frac{\sum_{i=1}^{n} ||t_i||}{n}$$

This is similar to `mallet`'s "tokens" metric which counts the number of word tokens in a topic [7]. The intuitions stay the same though: Larger topics tend to be more understandable and useful than overly specific single-document "topics". Such small topics do not offer many benefits as they are not clustering many documents, and not very reliable.

Topic Count The number of topics in the model.

$$\text{topic\_count} = ||T|| = n$$

Similar to the topic size metric, the topic count enables users to ensure that not too many small, specific topics are being generated instead of more useful hierarchical aggregations. Especially in situations where the IHTM algorithm does not find suitable insert positions, it tends to add the documents as new topics. This does not make for a good topic model. A quickly increasing topic count is a sign for users to check if topics could be merged into larger ones, also facilitating the search of insert positions for the algorithm for all following documents.

Speaker Count Calculates the average number of speakers present in the documents of any topic. Let $s(d)$ a function that returns the name of the speaker or author of a given document d.

$$\text{speaker\_count} = \frac{\sum_{t \in T} ||\{s(d) \mid d \in t\}||}{n}$$

This metric records the information that is also shown in the topic node pie charts in the visual analytics interface. It shows users whether topics are being primarily created based on the content of the documents, or based on speaker-specific language. Such speaker-specific language could be the frequent use of certain words or certain parts of speech. In the context of a debate, it might also be the case that some speakers ignore questions about a specific topic, and choose to talk about different things instead, which would lead to a low average speaker count.

2 Speculative Execution

In this section we give detailed information on both the available strategies for speculative execution and the implemented triggers from the model quality monitor component. While seven strategies are currently implemented, our system is modular and additional strategies can be easily added.

The Model Quality Monitor (MQM) marks on user-selected metric as "tracked", meaning that it will be highlighted in the timeline visualization. Additionally, it is used by the MQM in one optimization strategy and some speculative execution triggers instead of the entire set of metrics shown in the timeline. We call this set the "visible metrics" in the remainder of this section.

2.1 Optimization Strategies

To visualise the effects of each strategy we provide small sample figures. The tree on the left represents the model on which the speculative optimisation was triggered, while the tree on the right has been optimised by the respective strategy. To keep the examples minimal, different trees are used for each strategy. Nodes that have been moved between the two trees are connected with a dashed or dotted line. The dashed line signifies that the respective node was moved to a predetermined position in the new tree. Opposed to that, the dotted line shows that the node has been reinserted using the original IHTM algorithm's insert routine, which has determined this new location based on the *cosine similarity*. Deleted nodes are crossed out with a red line in the left tree, while new inner nodes that have been added through the optimisation are highlighted with a thicker border in the right tree.

The seven strategies listed in Table 2 are currently implemented and will be presented in more detail in the following.

Strategy	Description
Combine Similar Topics	Combines the most similar topics by moving them to a common parent node.
Reinsert Worst Topics	Identifies the "worst" topic under the currently tracked metric and redistributes its documents.
Reinsert Small Topics	Reinserts all documents belonging to topics with three or less documents.
Reinsert Outlier Nodes	Identifies and reinserts "outlier nodes" with the same algorithm used when creating the squiggly lines in the visualisation.
Split Largest Topic	Splits the largest topic (by number of documents) into two new topics.
Remove Topic Chains	Identifies "topic chains", a clustering artefact, and removes the chains by compacting the tree.
Default IHTM	The original IHTM algorithm serves as a baseline to compare other strategies against.

Table 2: The currently available speculative execution strategies: six optimisation strategies and the default IHTM algorithm as a baseline.

Combine Similar Topics This strategy finds the most similar topics t_1 and t_2 by calculating the pairwise similarity between the respective term frequency vectors of all topics. If two topics have been identified, a new, empty topic t_n is created at the top level. Both tree nodes of the topics t_1 and t_2 are then moved to t_n as its only two children. The effect of this operation is equivalent to the "join" operation presented by Duo *et al.* in their framework "HierarchicalTopics" [2].

This strategy can be employed when the IHTM algorithm is creating many highly specialised topics, but users are interested in a broader overview of the data.

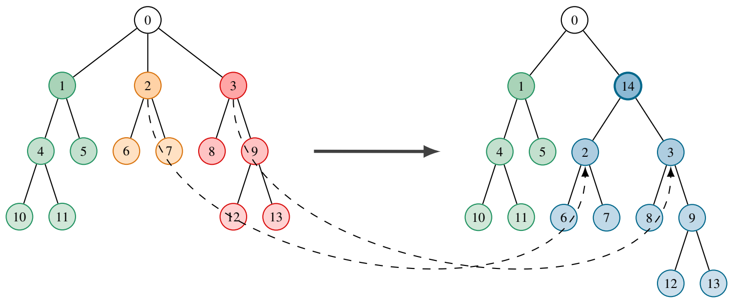

Figure 1: Effect of the "Combine Similar Topics" strategy. The topics 2 and 3 are determined to be the most similar, and are moved to a new common topic with the number 14.

Reinsert Worst Topic By evaluating the currently tracked metric for all topics, this strategy selects the topic with the worst score. If the tracked metric is either "Topic Count" or "Speaker Count" the strategy is unable to identify the worst topic and terminates without performing any modifications. In the first case, no topic can be meaningfully selected as the worst. In the second case, it is unclear whether a higher or lower number of speakers per topic is preferable, as this depends on the use-case and the corpus.

If a topic has been identified, all of its documents are removed from the model and reinserted by the same algorithm that was used for the initial insertion. The remaining old, empty inner nodes are deleted from the model. As the topic model might have evolved since the now removed and reinserted documents were first added, the algorithm will determine a different insert position for the reinserted documents if a better fitting topic is available now.

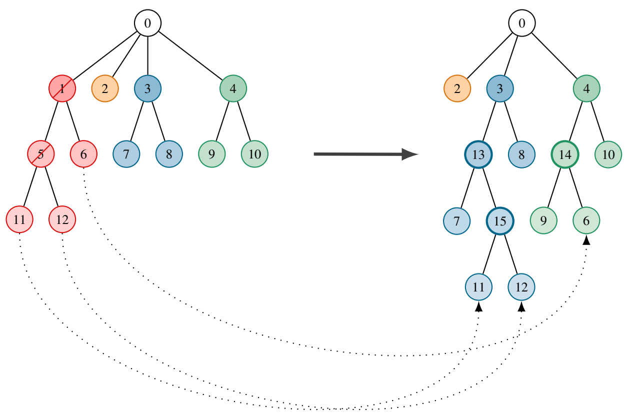

Figure 2: Effect of the “Reinsert Worst Topics” strategy. Topic 1 is determined to be the “worst” topic under the currently tracked metric, and its documents are redistributed.

It is important to note that all reinserted documents can be inserted into different topics during this process, leading to the complete dissolution of a previously existing topic. However, it is also possible that the same subtree that was just removed is recreated. Such a recreation will happen whenever the algorithm cannot identify better positions for the documents than the ones they already occupied before.

As the tracked metric can be selected by the users, this strategy enables them to automatically optimise the topic model for any given quality metric.

Reinsert Small Topics Many small topics with few documents can be an artefact of the cold start of the algorithm or a very loosely coupled corpus. This strategy finds all topics with three or fewer documents and removes them from the model. All removed documents are reinserted with the default IHTM algorithm, while all removed inner nodes are deleted.

If fewer than three topics would remain in the model after removing all identified small topics, the strategy terminates without performing any optimisation. This additional check prevents the strategy from deleting all topics at the beginning of the modelling process. Here, small topics are normal because not enough documents have been inserted into the model, yet.

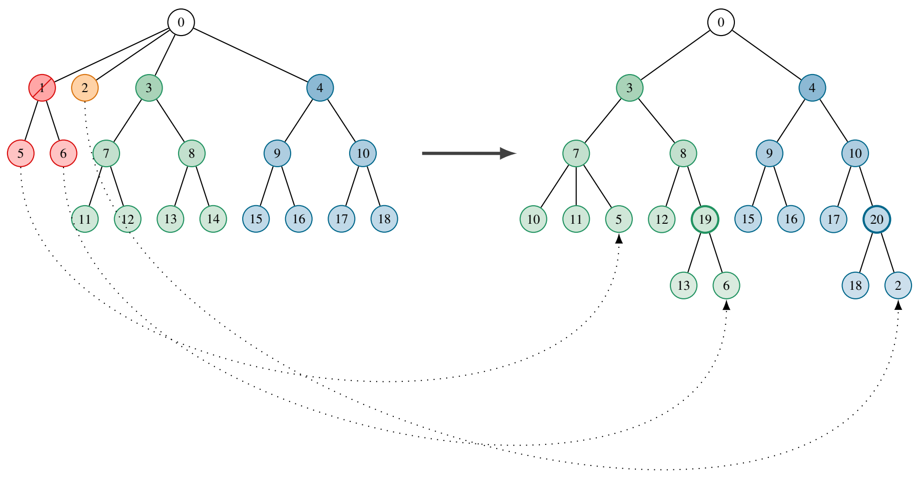

Figure 3: Effect of the “Reinsert Small Topics” strategy. The topics 1 and 2 contain less than three documents and are reinserted.

This strategy is used to move documents to their correct topics that have only been introduced after the original insertion of the now reinserted document. Additionally, it can lead to multiple small topics being combined into one larger one.

Reinsert Outlier Nodes While most other strategies in this list only operate on topic nodes (i.e. the roots of subtrees representing a topic), this strategy can target any inner node of the topic tree. It searches outlier nodes based on the variance of pairwise similarities between sibling nodes. First, the pairwise similarity between all children of a node is determined. For each child, we then assume that it is the outlier, remove it from the collection, and recalculate the variance of the pairwise similarities between the remaining children. The actual outlier is that child for which we determined the lowest variance while leaving it out. If this particular node is left out, all of its remaining siblings have, on average, the most similar word vectors, making the one that has been left out the outlier. This is the same algorithm that is used to generate the squiggly lines in our topic tree visualization. The only difference is its application to all inner nodes instead of the limitation to topic nodes.

The determined outlier node and its subtree are deleted from the tree. Afterwards, all documents that were just deleted are reinserted using the original IHTM insert algorithm, while old inner nodes are discarded.

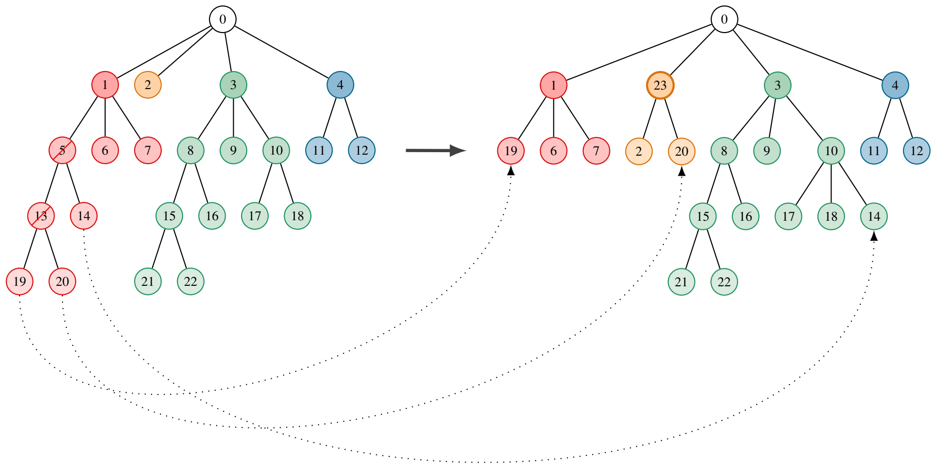

Figure 4: Effect of the “Reinsert Outlier Node” strategy. Node 5 has been determined as the outlier and has been reinserted. After the reinsertion document 19 is still part of topic 1.

While this strategy will occasionally reinsert entire topics, our experience after using the system indicates that it mostly removes subtrees after two topics have been incorrectly combined.

Split Largest Topic The largest topic t_l is determined by evaluating the number of documents in each topic. Two new topics t_1 and t_2 are created as the “split targets” and are initialized with the first and last document that was inserted into t_l, respectively. Then, the remaining documents from t_l are iteratively assigned to one of the two new topics in their original order of insertion into the topic. To determine the correct topic to insert a document into, this strategy simply calculates the *cosine similarity* between the term frequency vectors of the document and the topic. Note that the term frequency vector of the topic changes after every insert as it is the sum of all document descriptors contained in the topic. Using this approach over the default IHTM insertion algorithm guarantees that the topic is split into two new topics and that its documents are not potentially distributed over all other available topics. If such behaviour is desired, the “Reinsert Worst Topic” strategy should be applied instead.

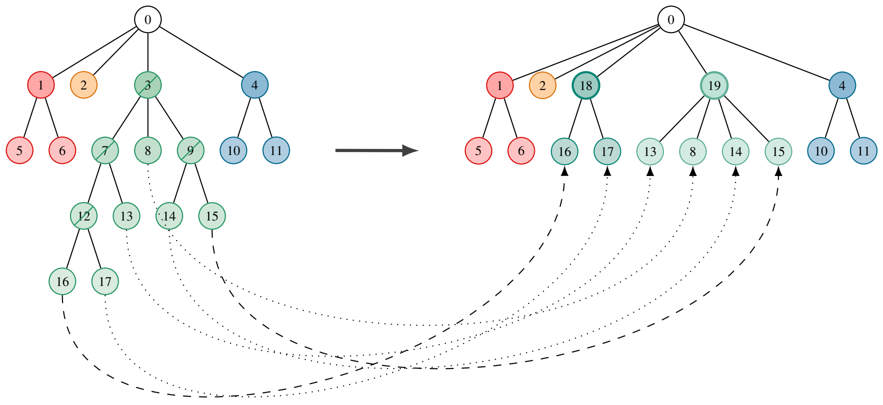

Figure 5: Effect of the “Split Largest Topic” strategy. Topic 3 has been split into the two new topics 18 and 19. Splitting a topic flattens its hierarchy.

One disadvantage of this strategy is that the hierarchy in the previously existing topic is flattened. This should be addressed in future work while being mindful of not reintroducing superfluous topic hierarchies.

Remove Topic Chains This strategy identifies “topic chains” as binary subtrees with a minimum height of two, excluding the root. Such chains are insertion artefacts of the IHTM algorithm and do not convey any meaningful hierarchy information. Applying this strategy deletes all inner nodes of all identified chains and changes all leaf nodes to be children of the (subtree) root.

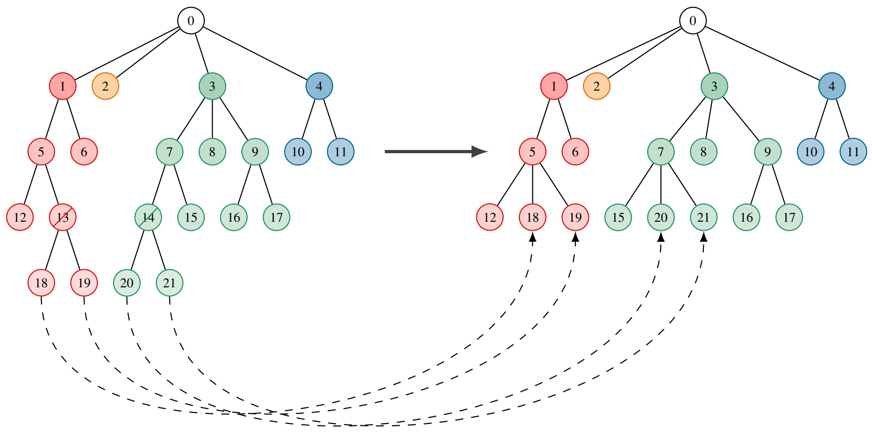

Figure 6: Effect of the “Remove Topic Chains” strategy. The nodes 5 and 7 are identified as “chain parents”. Their document nodes are moved to be direct children instead.

While this strategy influences metrics such as the tree compactness or the overall branching factor, it does not change the values of coherence, certainty, or any other metrics that are calculated based on the documents contained in a topic. This is because documents are only being moved within their topic, and neither the topic descriptor nor the topics term frequency vector change. The result is, however, a much more compact and comprehensible topic.

Default IHTM This strategy does not run any optimisation, and merely runs the default IHTM algorithm inserting the next ten documents. It is useful as a baseline to compare the effects of the other strategies against or can be selected in cases where users do not want to apply the results of any of the more specific optimisation strategies.

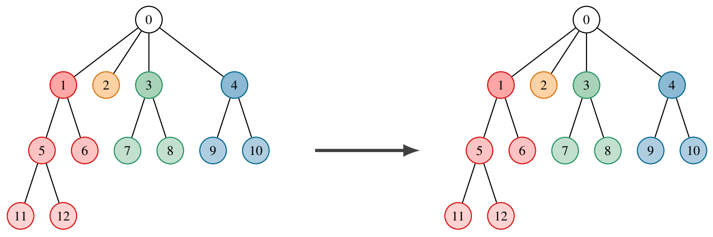

Figure 7: Effect of the “Default IHTM” strategy. As no optimisations are performed, this strategy acts as the baseline to compare the effects of other strategies against.

2.2 Triggers

To trigger the speculative execution automatically, we have implemented the model quality monitor. It contains the following sixteen triggers that can be selected by users based on their data.

To test the quality of the triggers, we visualized the topic modeling process of both the 20news corpus and the presidential debate data using our system. After every inserted document, we evaluated the trigger conditions for all implemented triggers and logged the respective result. At the same time, we observed the topic model visualization and employed our knowledge of the two corpora to determine whether a document insert made sense, or whether significant errors had happened that an optimization strategy could have improved. Typical situations that led to manual triggering were the wrong creation of new, duplicate topics or (repeated) inserts of similar documents in the wrong topic. While we tested whether a trigger fired or not, we never started a speculative execution cycle but continued running the default IHTM algorithm to limit the number of variables in our experiment. The “tracked metric” had been set to coherence. The “visible metrics” were coherence, certainty, compactness, topic size and topic count.

For this evaluation, we calculated precision, recall and f-score values for all trigger strategies. While we had our observations as ground truth data to compare against, we did not expect the triggers to start a speculative execution at the exact time that we thought it necessary. Instead, we introduce an ε-environment for any point in time and consider a positive trigger outcome a true positive if we would have manually triggered the speculative execution within the last or following ε document inserts. For this evaluation we chose an ε-values 2. Introducing such an ε value also increases the performance difference between better and worse trigger strategies, making them easier to identify. For example, the f-score for the “TM” trigger indicated worse performance than the “MDI” trigger at $\varepsilon = 0$. However, the latter of the two has triggered after every document insertion and has produced the highest number of false positives in our tests. For $\varepsilon = 2$, there is a large performance difference between the two strategies, with “TM” now showing a significant improvement over a naive “trigger-always” strategy of “MDI”.

We argue that the use of an ε-environment in this evaluation is valid as the automated trigger system is intended for users that do not want to observe the entirety of the modelling process closely. For them, it will be irrelevant whether the trigger strategy started a speculative execution one document insert earlier or later, as long as it identified the deteriorating model quality and started the necessary optimisations.

The available triggers are presented in the following table:

Name	Description	F-Score
Tracked Metric Decrease (TM)	This trigger fires if the **tracked quality metric** worsens by more than five percent between two document inserts. Depending on the metric this means an increase or decrease of five percent.	47.1
Majority of Visible Metrics Decreases **(MVM)**	Fires if the **majority of metrics** visible to the user in the timeline worsens by more than five percent between two document inserts.	59.3
Any Visible Metric Decreases	Fires if **any of the metrics** the users selected as visible in the timeline worsens by more than five between to document inserts.	61.5
All Visible Metrics Decrease	Fires if **all metrics** visible to the user in the timeline worsen by more than five percent between to document inserts.	0
Metric Difference Increase (MDI)	Fires if the pairwise distance between any two metrics increases	16.7
Tracked Metric Below its Ten Step Average	Fires if the **tracked metric** has fallen **below its moving average** calculated over the metric values for the **last ten document inserts**.	16.5
Majority of Visible Metrics below their Ten Step Average	Fires if the **majority of visible metrics** has fallen **below their moving averages** calculated over the metric values for the **last ten document inserts**.	17.0
Any Visible Metric below its Ten Step Average	Fires if **any visible metric** has fallen **below its moving average** calculated over the metric values for the **last ten document inserts**.	28.6
All Visible Metrics below their 25 Step Average	Fires if **all visible metrics** have fallen **below their moving averages** calculated over the metric values for the **last 25 document inserts**.	0
Tracked Metric Below its 25 Step Average	Fires if the **tracked metric** has fallen **below its moving average** calculated over the metric values for the **last 25 document inserts**.	13.3
Majority of Visible Metrics below their 25 Step Average	Fires if the **majority of visible metrics** has fallen **below their moving averages** calculated over the metric values for the **last 25 document inserts**.	12.7

continued on the next page

Table 3 continued from previous page

Name	Description	F-Score
Any Visible Metric below its 25 Step Average	Fires if **any visible metric** has fallen **below its moving average** calculated over the metric values for the **last 25 document inserts**.	14.0
All Visible Metrics below their 25 Step Average	Fires if **all visible metrics** have fallen **below their moving averages** calculated over the metric values for the **last 25 document inserts**.	0
Rapidly Falling Slope for Any Visible Metric **(RFa)**	Fires if **any visible metric** has worsened for at least two document inserts in a row, with the last insert leading to a bigger decrease in measured quality than the previous one.	57.1
Rapidly Falling Slope for Majority of Visible Metrics	Fires if the **majority of visible metrics** has worsened for at least two document inserts in a row, with the last insert leading to a bigger decrease in quality measured by the respective metric than the previous one.	0
Rapidly Falling Slope for All Visible Metrics	Fires if **all visible metrics** have worsened for at least two document inserts in a row, with the last insert leading to a bigger decrease in quality measured by the respective metric than the previous one.	0
RFa & MVM	**Combination of "Majority of Visible Metrics Decreases" and "Rapidly Falling Slope for Any Visible Metric"**.	**70.4**

Table 3: The implemented speculative execution triggers.

References

[1] G. Bouma. Normalized (Pointwise) Mutual Information in Collocation Extraction. *Proceedings of the GSCL*, pages 31–40, 1 2009.

[2] W. Dou, L. Yu, X. Wang, Z. Ma, and W. Ribarsky. HierarchicalTopics: Visually exploring large text collections using topic hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2002–2011, 2013.

[3] M. El-Assady. Incremental Hierarchical Topic Modeling for Multi-Party Conversation Analysis, 2015.

[4] M. El-Assady, R. Sevastjanova, F. Sperrle, D. Keim, and C. Collins. Progressive Learning of Topic Modeling Parameters: A Visual Analytics Framework. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):382–391, 2018.

[5] G. Huang, C. Guo, M. J. Kusner, Y. Sun, F. Sha, and K. Q. Weinberger. Supervised Word Movers Distance. In *Advances in Neural Information Processing Systems*, pages 4862–4870, 2016.

[6] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From Word Embeddings To Document Distances. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 957–966, 2015.

[7] A. K. McCallum and D. Mimno. MALLET: A Machine Learning for Language Toolkit, 2002.

[8] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum. Optimizing semantic coherence in topic models. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272, 2011.

[9] H. Misra, O. Cappé, and F. Yvon. Using LDA to detect semantically incoherent documents. *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 41–48, 2008.

[10] S. I. Nikolenko, S. Koltcov, and O. Koltsova. Topic modelling for qualitative studies. *Journal of Information Science*, 43(1):88–102, 2017.

[11] K. Stevens, P. Kegelmeyer, D. Andrzejewski, and D. Buttler. Exploring Topic Coherence over Many Models and Many Topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 952–961, 2012.