Viscovery: Trend Tracking in Opinion Forums based on Dynamic Topic Models

Ignacio Espinoza, Marcelo Mendoza, Pablo Ortega, Daniel Rivera, Fernanda Weiss
Department of Informatics
Universidad Técnica Federico Santa María
Santiago, Chile
ignacio.espinozav@sansano.usm.cl
marcelo.mendoza@usm.cl
pablo.ortega@usm.cl
daniel.riverat@alumnos.usm.cl
fernanda.weiss.13@sansano.usm.cl

Journal of Digital Information Management

ABSTRACT: Backgrounds: *Opinions in forums and social networks are released by millions of people due to the increasing number of users that use Web 2.0 platforms to opine about brands and organizations. For enterprises or government agencies it is almost impossible to track what people say producing a gap between user needs/ expectations and organizations actions.*

Methods: *To bridge this gap we create Viscovery, a platform for opinion summarizing and trend tracking that is able to analyze a stream of opinions recovered from forums. To do this we use dynamic topic models, allowing to uncover the hidden structure of topics behind opinions, characterizing vocabulary dynamics. We extend dynamic topic models for incremental learning, a key aspect needed in Viscovery for model updating in near-real time. In addition, we include in Viscovery sentiment analysis, allowing to separate positive/negative words for a specific topic at different levels of granularity.*

Results: *Viscovery allows to visualize representative opinions and terms in each topic. At a coarse level of granularity, the dynamic of the topics can be analyzed using a 2D topic embedding, suggesting longitudinal topic merging or segmentation.*

Conclusions: *In this paper we report our experience developing this platform, sharing lessons learned and opportunities that arise from the use of sentiment analysis and topic modeling in real world applications.*

Subject Categories and Descriptors
H.3 [INFORMATION STORAGE AND RETRIEVAL]; Systems and Software

General Terms: Search Interfaces, Topic Modeling

Keywords: Opinion mining platforms, Data visualization, Search tools

Received: 14 August 2018; Revised 24 October 2018, Accepted 4 November 2018

Review Metrics: Review Scale - 0/6, Review Score - 4.8, Inter-Reviewer Consistency - 83.5%

DOI: 10.6025/jdim/2019/17/1/13-24

1. Introduction

The emergence of the Web 2.0 has allowed that millions of users can send posts and opinions about celebrities, institutions, organizations and brands. As the volume of opinions in forums and blogs increases, the need to develop effective platforms for opinion search has become urgent. In the stream of opinions, trend tracking is a key building block of this kind of platforms, allowing to describe what users expect about institutions/ organizations and how opinion trends evolve over time.

Effective tools for opinion browsing need to incorporate opinion aggregation functionalities, being relevant to obtain descriptions of each trend. In addition, the sentiment orientation of opinions w.r.t. named entities lights up how users act/react in front of a given organization. Sentiment analysis methods are helpful in this task.

As the volume of opinions is huge, the need to develop effective aggregation methods over opinions is the key building block of any opinion trend platform. Opinion clustering is a way to aggregate opinions. Using hard clustering algorithms each opinion can be assigned to a single class. However, documents achieve a best description by modeling its content with a mixture of topics, where each topic is defined as a probability distribution over words. In this way, opinions belong to several topics with different degrees of membership. This is the reason why documents are modeled using mixed membership models, and in specific Latent Dirichlet Allocation (LDA) [3], allowing to uncover the hidden structure of topics behind a corpus. LDA has made improvements in information retrieval tasks [13] outperforming standard text clustering algorithms being the state-of-theart method for document aggregation.

In Chile, the National Agency of Consumers 1 centralize complaints about brands and their products. As it is almost impossible to follow each complaint, consumers may be disappointed due to the slow response of the Agency to their needs. To bridge this gap we created Viscovery, a platform for opinion aggregation and tracking that allow to browse a huge volume of opinions in a few minutes. The core of Viscovery is based on Dynamic Topic Models (DTM) [2], an extension of LDA [3] capable of model a time sliced corpus, being able to estimate dependencies between vocabularies across time slices. To create Viscovery, we had to develop an incremental learning component able to update a model with new opinions. In addition, we include in Viscovery sentiment analysis. Sentiment analysis allows to distinguish between subjective/neutral terms in each distribution of words, enlightening how consumers opine about brands and products. To include sentiment analysis in DTM we explore a simple approach based on aggregation, using lexical analysis at opinion level and conducting sentiment aggregations at topic and document level. A third element included in Viscovery is topic embedding. Using a time sliced 2D topic embedding, topic merging and topic segmentation are suggested. Dynamics across topics are very interesting for the analysis, and it is a promising characteristic of Viscovery that allow practitioners to understand how topics evolve. Specific contributions of the paper are:

• A scalable implementation of DTM for online training updating model parameters when new opinions come to the platform.

• A simple way to incorporate sentiment analysis into DTM, allowing to explore neutral/subjective words at different levels of aggregation.

• A topic visualization tool that works with a time sliced 2D topic embedding, allowing to visualize how topics evolve over time.

The rest of the paper is organized as follows. In Section 2 we review related work on topic models and sentiment analysis. Incremental learning on DTM is presented in Section 3 and browsable sentiment analysis is discussed in Section 4. Section 5 presents the architecture of Viscovery. Implementation issues are discussed in Section **??**. Viscovery data slices are presented in Section 6 and finally we conclude in Section 7 giving conclusions and discussing future work.

2. Related Work

Topic Models. Main efforts on topic models start with probabilistic Latent Semantic Analysis [10] (pLSA), an aspect model for text developed using topic mixtures. This approach decomposes a corpus of documents across terms introducing latent variables, decoupling terms and documents with topic mixtures. Model fitting was conducted using the Expectation-Maximization algorithm (EM) [5] suitable to matrix completion with incomplete data. As the term-space is a high dimensional feature space, pLSA needs a high amount of data to perform well. As in general, text data is sparse, pLSA tends to over-fit limiting generalization capabilities. To tackle this problem, Blei *et al.* [3] introduced Dirichlet priors on vocabulary and document topic proportions. Using smoothing these models addressed the over-fitting limitations of pLSA. This kind of models, known as Latent Dirichlet Allocation (LDA) were firstly fitted using variational EM (VEM), an extension of the EM algorithm that successfully handle incomplete data with distributional priors. Later, Griffiths and Steyvers [8] explored Gibbs sampling for LDA model fitting, reducing the number of iterations until convergence. Gibbs sampling is the standard method used for LDA model fitting until today because its fast convergence does not affect the quality of the estimated models. Dynamic Topic Models (DTM) [2] was introduced to deal with vocabulary dynamics. DTM works over a corpus with time-stamps, whilst model fitting is conducted using time slices of the corpus. Temporal dependencies across vocabularies are modeled using Kalman filtering, allowing to detect changes in descriptive words along different corpus slices. The inclusion of Kalman filtering in LDA for text dynamics involves additional computational costs in model fitting, slowing convergence. Despite computational costs involved in model fitting, DTM can successfully handle text dynamics. Topics modeling has many applications standing out among them document clustering [16], book recommendation [19], metadata enrichment [17], social bookmark recommendation [9], and Newspaper analysis [21].

[1]Sernac: http://www.sernac.cl

Sentiment Analysis and Topic Modeling. Sentiment analysis has shown to be a useful tool for the processing of subjective information [4]. In specific, the use of sentiment analysis and topic modeling shows advances. A topic generative model for sentences with polarity was proposed by Eguchi and Lavrenko [6]. The model distinguishes between neutral words and sentiment words using a random binary variable that controls the membership of each word to each one of the topics. As documents can be generated from sentiment or topic words, each sentence achieves a polarity orientation calculated in terms of the number of sentiment words that contains. Dirichlet smoothing was used on topic and sentiment word distributions to avoid over-fitting. The performance of the model in information retrieval is tested inferring topic and sentiment orientation of each query showing that the proposal is feasible. Mei *et al.* [15] proposed Topic Sentiment Mixture (TSM), a sentiment topic model with a two tier mixture of vocabularies to produce sentiment oriented sentences in a corpus. A first tier of the model is composed by neutral term distributions (one per each topic) and two additional term distributions for positive and negative words. Then, each topic can be produced by a mixture of these vocabularies defined from document proportions. The model is non-parametric (no distributional priors were used) and model fitting was conducted using the EM algorithm. TSM can be considered as an extension of pLSA to sentiment analysis being the main difference the split conducted over the vocabulary to distinguish between factual/subjective sentences. The Joint Sentiment Topic model (JST) based on LDA was proposed by Lin and He [14]. Term distributions were sampled over a simplex over terms cross polarities, then the generative model drawn topic proportions conditioned on each polarity. In this way, words can be drawn by topics polarities distributions, producing words by the joint effect of topics and polarities in the document. As a consequence, sentiment coverage at document level can be directly estimated by the model. JST is able to successfully address the sentiment classification task at document level. An extension of JST was proposed by Jo and Oh [12], who introduced Aspect and Sentiment Unification (ASUM). As JST, ASUM jointly models sentiments and topics, being topic proportions conditioned on polarities, with vocabularies at topic level per each sentiment orientation. However, ASUM models sentiment at sentence level, with words conditioned at a single topic per sentence. Results on sentiment classification shows that ASUM outperforms JST and it comes close to supervised methods whilst ASUM does not require labels for model fitting. The state of the art shows that main efforts on sentiment topic modeling are focused on static models, discarding vocabulary dynamics. As the core of Viscovery is DTM, we will need to use a different approach to include sentiment analysis into dynamic topic models. We will show in Section 4 how we use sentiment analysis at sentence level to conduct aggregation at different levels of granularities over DTM.

3. Incremental Learning for Dynamic Topic Models

3.1 Dynamic Topic Models

A set of latent variables can be introduced to model the relationships between terms and documents in a corpus. Formally, let $d \in \mathscr{D} = \{d_1, d_2, \ldots, d_N\}$ and $w \in \mathscr{W} = \{w_1, w_2, \ldots, w_M\}$ be random variables representing documents and terms, respectively. A set of random variables $z \in \mathscr{Z} = \{z_1, z_2, \ldots, z_k\}$ can be introduced to model the joint probability of documents and terms, producing a mixed membership model expressed as follows:

$$P(w|d) = \sum_{z \in Z} P(w|z) \cdot P(z|d) \qquad (1)$$

Using the Bayes rule to invert the conditional probability $P(z|d)$, we obtain an expression of the joint probability conditioned to the model parameters:

$$P(w,d) = \sum_{z \in Z} P(w|z) \cdot P(d|z) \cdot P(z) \qquad (2)$$

The equation 2 is known as the generative formulation of the topic model of the corpus.

Topic models based on Dirichlet allocation require two Dirichlet distributions. A first one generates topic proportions for each document and a second one generates terms conditioned on document topics proportions. Specifically, a Dirichlet k-dimensional random variable q takes values in a k-1 simplex $(0 \le \theta_i \le 1, \sum_{i=1}^{k} \theta_i = 1)$, where its density function is defined by:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \quad \theta_1^{\alpha_1} \cdot \ldots \cdot \theta_k^{\alpha_k}, \qquad (3)$$

and $\{\alpha_1, \ldots, \alpha_k\}$ corresponds to the distributional parameters, $\alpha_i > 0$. Then, equation 2 is expanded using Dirichlet priors:

$$P(W,d) = \prod_{n=1}^{M} P(w_n|z_n, \beta) \cdot P(z_n|\theta_d) \cdot P(\theta_d|\alpha) \qquad (4)$$

In equation 4, θ_d indicates the proportion of topics in d. Then, z_n is conditioned on β and represents the sampling probability of w_n on d. Note that α and β are the distributional parameters of the Dirichlet density functions. Usually they are consigned as hyperparameters to make a difference with model parameters. It is common to make an assumption of density symmetry for hyper-parameters, that is $\alpha_1 = \ldots = \alpha_k = \alpha$ and $\beta_1 = \ldots = \beta_k = \beta$. The values α, β control the level of smoothness/sharpness of the density functions around the centroid of the simplex.

To model a time sliced corpus, Blei and Lafferty [2] introduced dynamic topic models (DTM). DTM is based on the static Latent Dirichlet Allocation model and use the mean parameterization of the multinomial topic distribution. The idea behind DTM is to use the mean parameterization of the topics to introduce mean chaining, being possible to model time dependencies over time. To chain topics over time, DTM employs the chain of mean parameters introducing Gaussian noise, modeling

uncertainty over time slices. Let $\beta_{t;k}$ be the k-th topic in the time slice t and let π be the mean parameter of the topic. Note that the *i*-th component of $\beta_{t;k}$ is given by $\beta_i = \log\left(\frac{\pi_i}{\pi_V}\right)$. As π_i represents the expected value of w_i and π_v is the expected value of a random chosen word over the whole vocabulary V, the fraction $\frac{\pi_i}{\pi_V}$ is the odd of w_i over V and then β_i corresponds to the *logit* function for w_i over V. As is known, a zero variation over V achieves a zero value in the *logit* function. Positive or negative deviations of w_i in V achieves positive or negative values in [-1, +1], respectively. Then, $\beta_{t,k}$ can be chained in a state space of parameters that evolves with Gaussian noise:

$$\beta_{t,k}|\beta_{t-1,k} \sim \mathcal{N}(\beta_{t-1,k}, \sigma^2\mathscr{I}) \quad (5)$$

Topic proportions are also chained in DTM, using mean parametrization over θ:

$$\theta_t|\theta_{t-1} \sim \mathcal{N}(\alpha_{t-1}, \delta^2\mathscr{I}) \quad (6)$$

Time chaining does not affect model expressiveness. In fact, the decomposition of the joint distribution of words and documents in a corpus remains the same, except for the fact that both Dirichlet distributions (on topic proportions and terms) are conditioned on the Dirichlet distributions of the previous time slice:

$$P(W,d,t) = \prod_{n=1}^{M} P(w_n|z_n,\beta_t)\cdot P(z_n|\theta_{d,t})\cdot P(\theta_{d,t}|\alpha) \quad (7)$$

Model estimation has some drawbacks under these assumptions. Posterior inference (model estimation of parameters conditioned on observed variables) is intractable due to the non conjugacy of Gaussians and multinomial distributions. Blei and Lafferty explored variational methods for posterior inference, discarding stochastic simulation (e.g. Gibbs sampling) due to computational difficulties inherent in the non conjugacy of Gaussians. To retain the sequential structure of topics over time, DTM fits a dynamic model with Gaussian variational observations ($\hat{\beta}_{k,1},\ldots,\hat{\beta}_{k,t},\ldots,\hat{\beta}_{k,T}$), fitting these parameters to minimize the Kullback-Leibler divergence between the model resulting posterior and the true posterior. To mimic Gaussian variational observations, DTM uses Kalman filtering, which enables the use of backward-forward calculations in a linear state space model. Analogously, topic proportions $\theta_{t,d}$ are conditioned on free Dirichlet parameters $\gamma_{t,d}$ and topic indexes $z_{t,d,n}$ are conditioned on free multinomial parameters $\phi_{t,d,n}$:

$$q(\beta_{k,1},\ldots,\beta_{k,T}|\hat{\beta}_{k,1},\ldots,\hat{\beta}_{k,T}), \quad (Kalman) \quad (8)$$

$$q(\theta_{t,d}|\gamma_{t,d}), \quad (Dirichlet) \quad (9)$$

$$q(z_{t,d,n}|\phi_{t,d,n}), \quad (Multinimial) \quad (10)$$

Forward-backward calculations on Kalman parameters allows to estimate posterior mean and variance parameters (m_t and V_t) in terms of Gaussian parameters (σ^2) over topics. As time sliced topics β_t are conditioned on the immediate past time sliced topic β_t–1, and both are related by a Gaussian of σ parameter, the variational state space model $\hat{\beta}_t$ is conditioned on β_t and both are related by a Gaussian of $\hat{v}_t$ parameter. Then, in forward calculations, posterior mean and variance parameters (m_t and V_t) are calculated from σ and from the variational parameters $\hat{\beta}_t$ and $\hat{\sigma}_t$. In backward calculations the marginal mean $\tilde{m}_{t-1}$ and variance $\tilde{v}_{t-1}$ of β_{t-1} depends on posterior mean m_{t-1} and variance V_{t-1}, σ and the one-step ahead marginal mean $\tilde{m}_t$ and variance $\tilde{V}_t$. Forward calculations use initial conditions m_0 and V_0 and backward calculations use initial conditions $\tilde{m}_T = m_T$ and $\tilde{V}_T = V_T$. The rest of the parameters are estimated using variational expectation maximization (VEM) as it was proposed in the original posterior inference algorithm of LDA [3].

3.2 Variational Inference Algorithm

To estimate model parameters, DTM works using VEM and variational Kalman filtering in a tandem. The inference algorithm starts initializing Kalman parameters using the LDA static VEM inference algorithm over the whole corpus, discarding time-stamps. As an output of this process DTM obtains Kalman variational parameters ($\hat{\beta}_i$). Forward calculations are conducted to estimate posterior means and variances $m_t = \mathbb{E}(\beta_t|\hat{\beta}_{1:t})$ and $V_t = \mathbb{E}((\beta_t - m)^2|\hat{\beta}_{1:t})$ with initial conditions $m_0 = 0$ and $V_0 = \sigma^2 \cdot \mathrm{E}{+}03$. Backward recurrences are used to estimate marginal means and variances $\tilde{m}_{t-1} = \mathbb{E}(\beta_{t-1}|\hat{\beta}_{1:T})$ and $\tilde{V}_{t-1} = \mathbb{E}((\beta_{t-1} - \tilde{m}_{t-1})^2|\hat{\beta}_{1:T})$ with initial conditions $\tilde{m}_T = m_T$ and $\tilde{V}_T = V_T$.

Likelihood bound variables $\zeta_t = \sum_w e^{\tilde{m}_{tw}+0.5\cdot\tilde{V}_{tw}}$ are computed for each topic in each time slice and $\beta_{t,k,n}$ for each topic, term and time slice in the corpus.

After the initialization step, the inference algorithm runs the EM algorithm. The E-step uses the static LDA VEM inference algorithm, at document level, in chronological order according to document time-stamps. Then, for each document a free Dirichlet parameter $\gamma_{t,d}$ is obtained, and for each word in each document a multinomial parameter $\phi_{t,d,n}$ is obtained. The E-step iterates until convergence following the LDA VEM convergence criteria. Then, the M-step runs bounding topic likelihoods. The process repeats the steps considered in the initialization process conditioned on $\phi_{t,d,n}$ model parameters. Forward and backward calculations are conducted estimating variational Kalman parameters and iterating until convergence following a topic likelihood criteria. At global level, E-step and M-step alternates until convergence, following a criteria that combines document and topic likelihoods.

3.3 Incremental Learning on DTM

A key aspect of Viscovery relies on incremental learning. As topics are used as opinion containers, the need to incorporate new opinions is a key aspect to keep information updated. To avoid the recalculation of the entire model, we extended DTM to allow incremental learning, updating the model to be consistent with new opinions but avoiding the recalculation of model parameters that

depends on previous time slices.

When a new batch of documents is fed into Viscovery, a set of unseen words may appears. Suppose that Q new words are appended by the new batch to the model and assume that the batch size (number of documents in the batch) is R. Let $\mathscr{W}^{\text{NEW}} = \{w_{M+1}, \ldots, w_{M+Q}\}$ be the set of new words and let $\mathscr{D}^{\text{NEW}} = \{d_{N+1}, \ldots, d_{N+R}\}$ be the new batch. A first set of new parameters is in dependence on $\mathscr{W}^{\text{NEW}}$ Rigs in dependence on W new and the previous slices $1:T$. The topic parameters included into the model are $\beta_{M+1,1:T}, \ldots, \beta_{M+Q,1:T}$. As these new words were unobserved on previous slices, we set these parameters using blong-tail, the value assigned by DTM to words in the Zipf long-tail of the model. In practice, by choosing at random any word from the tail of any topic, $\beta_{\text{LONG-TAIL}}$ achieves only small fluctuations (order 10^{-12}). Analogously, we set $\hat{\beta}_{M+1,1:T}, \ldots, \hat{\beta}_{M+Q,1:T}$ as $\hat{\beta}_{\text{LONG-TAIL}}$, the value assigned by DTM to words in the long-tail of the Kalman variational parameters. Then we fit a static LDA over the new batch to obtain initial values for $\beta_{1:Q,T+1}$ parameters (model parameters for the new batch over the whole vocabulary). Mean and variance parameters (variational and marginal) are calculated using the forward-backward procedure at one step (one step ahead for forward calculation and one step back for backward recursion). To avoid unnecessary computation costs, we discarded the recalculation of the entire chain of Kalman variational parameters, constraining the inference only to dependencies between batches in the slices T and $T+1$. The constrained forward-backward calculation produces estimations for mean and variance in the new batch, and values for likelihood bounds ζ_{T+1} and $\beta_{T+1,k,n}$.

Now we follow the EM procedure. Log likelihoods of topics and documents modeled in previous batches are retrieved to be included in the global likelihood criterion function used in the EM procedure. The E-step is conducted over the documents included in $\mathscr{D}^{\text{NEW}}$, obtaining estimates for $\gamma_{T+1,d}$ and $\phi_{T+1,d,n}$, $d \in \mathscr{D}^{\text{NEW}}$, likelihood bounds for each document. The E-step runs until convergence following the LDA VEM convergence criteria. The M-step runs bounding topic likelihoods. The process repeats the following cycle at word level for each document in the new batch: REPEAT: TOPIC BOUND EST $\rightarrow$ BATCH MODEL UPDATING $\rightarrow$ NEW TOPIC BOUND $\rightarrow$ CHECK CONVERGENCE. As the M-step runs over the last chain of DTM, the convergence is very fast and the overall convergence is also very fast. In the appendix we give more details about how our incremental algorithm guarantees that the log likelihood $\log p(d_{1:T})$ is bounded from below using the inequality of Jensen [2].

4. Browsable Sentiment Analysis

In this section we indicate how we produce a browsable sentiment analysis view of the data in Viscovery defining three levels of data aggregation. Sentiment analysis is a key aspect of opinion mining tools and in Viscovery is a salient aspect that helps users to distinguish between subjective/neutral information. As a base service, our Novaviz API uses VADER [11] for sentiment sentence tagging. VADER provides three sentiment scores at sentence level: positive ($\text{sc}_s(\oplus)$), negative ($\text{sc}_s(\ominus)$) and neutral ($\text{sc}_s(\odot)$) scores, where $\text{sc}_s(\oplus)+\text{sc}_s(\ominus)+\text{sc}_s(\odot) = 1$. We retrieve for each sentence in our corpus these three scores.

Document Level Aggregation. A first level of aggregation considered in Viscovery is the document level. As opinions can be compounded by a number of sentences, sentiment scores need to be aggregated at opinion level. Let d be a document indexed in Viscovery, and $s \in d$ the sentences that compound d, where $|d|$ is the number of sentences of d. Sentiment scores at document level are computed as follows:

$$\text{SC}_d(*) = \sum_{s \in d} \frac{\text{SC}_s(*)}{|d|}, \quad \text{with } * \in \{\oplus, \ominus, \odot\} \qquad (11)$$

Note that $\text{sc}_d(\oplus)+\text{sc}_d(\ominus)+\text{sc}_d(\odot) = 1$, as expected.

Topic Level Aggregation. A second level of aggregation considered in Viscovery is the topic level. As opinions are aggregated into topics, sentiment scores need to be aggregated at topic level to indicate the level of polarity of each topic. Let z be a LDA latent variable, and $P(d|z)$ the membership probability given by DTM and defined in Equation 2. Note that $P(*|z) = \sum_{d \in \mathscr{D}} P(*|d) \cdot P(d|z)$. For simplicity, we denote $P(*|z)$ by $\text{SC}_z(*)$. Then, sentiment scores at topic level are obtained as follows:

$$\text{SC}_z(*) = C_z \cdot \sum_{d \in \mathscr{D}} \text{SC}_d(*) \cdot P(d|z), \quad \text{with } * \in \{\oplus, \ominus, \odot\} \qquad (12)$$

where $C_z = \frac{1}{\sum_* \text{SC}_z(*)}$. Note that $\text{sc}_z(\oplus)+\text{sc}_z(\ominus)+\text{sc}_z(\odot) = 1$ as expected.

Term Level Aggregation. At a fine level of granularity Viscovery browses terms. To use sentiment analysis at term level, we need to estimate $P(*|w)$, denoted for simplicity by $sc_w(*)$. As $sc_w(*)$ can be expanded over latent variables by $\Sigma_{z \in Z} sc_z(*) \cdot P(z|w)$, using the Bayes rule on $P(z|w)$ we obtain $sc_w(*)$ as follows:

$$\text{SC}_w(*) = \sum_{z \in Z} \text{SC}_z(*) \cdot \frac{P(w|z) \cdot P(z)}{P(w)}, \quad \text{with } * \in \{\oplus, \ominus, \odot\} \qquad (13)$$

Note that $sc_w(\oplus)+sc_w(\ominus)+sc_w(\odot) = 1$, as expected.

Using Topics as Proxies of Polarities. Viscovery allows to browse opinions using topics. When a topic is picked in Viscovery, the sentiment view of the data can be projected to documents or terms. Then, topics can be used as proxies of polarities, as sentiment scores can be computed conditioned on topics. Intuitively, as some topics tend to lean towards specific polarities, the conditioning of sentiment scores to topics provides a fine granularity

analysis of the polarity spectra in a specific topic. To show sentiment scores conditioned on topics, we reuse the scores defined in equations 11-13. Sentiment scores at document level conditioned on topics are defined as follows:

$$\mathrm{SC}_d(*|z) = \left(\sum_{w \in \mathscr{W}} \mathrm{SC}_w(*) \cdot P(w|z) \right) \cdot P(z|d), * \in \{\oplus, \ominus, \odot\} \quad (14)$$

Analogously, sentiment scores at term level conditioned on topics are defined as follows:

$$\mathrm{SC}_w(*|z) = \left(\sum_{d \in \mathscr{D}} \mathrm{SC}_d(*) \cdot P(d|z) \right) \cdot P(z|w), * \in \{\oplus, \ominus, \odot\} \quad (15)$$

This simple way to aggregate scores from sentence sentiment scores allows us to use sentiment analysis on DTM.

5. Viscovery: Architecture and Search Interface

5.1 Architecture

In this section we discuss how we integrate different algorithms and tools to ingest opinions into Viscovery. We model different algorithms as micro services to develop a platform for trend tracking in opinion forums. A micro service architecture organizes the platform as a set of weakly coupled services where each service implements a set of encapsulated procedures. For example, a micro service in Viscovery corresponds to an indexer of opinionated tweets. Services in Viscovery are communicated using asynchronous protocols. We developed each service independently of the other. Indeed each micro service has its own database in order to be decoupled from other services.

To develop Viscovery we create a start-up named Novaviz. The idea behind Novaviz is to develop tools for text data management. To accomplish this purpose we develop the Novaviz API Gateway, a list of services and functionalities implemented in Python, requested by four components: a) Data Ingester, b) Data Preprocessor, c) Data Processor, and d) Indexer. For data visualization we use three libraries: a) DFR browser, b) Kibana, and c) D3. Visualization and processes are connected through micro services defined in the Novaviz API Gateway, as it is shown in the architectural diagram of Figure 1.

As Figure 1 shows, Viscovery is composed by the following components:

Data Ingester. This component is in charge of opinion scrapping from heterogeneous sources as Twitter and web forums (e.g. http://www.reclamos.cl). It calls services from the Novaviz API. Among the services requested the most important is web scrapping, that allows Viscovery to retrieve opinions from web page forums. For storage, this component interacts with Redis (https://redis.io/), an open source (BSD licensed) inmemory data structure store, used as a cache database to support this process.

Preprocessor. This component normalizes the text. It calls services from the Novaviz API such as stop words removal, caps normalization and punctuation removal. It allows Viscovery to create a vocabulary of keywords to describe opinions by content. For storage this component interacts with MongoDB (https://www.mongodb.com/), a database for document storage and retrieval.

Processor. This component is in charge of text analysis and it is the core component of Viscovery. It calls services from the Novaviz API as Dynamic Topic Models and Sentiment Analysis. For Dynamic Topic Models (DTM), the API wraps Gensim (http://radimrehurek.com/gensim/). Gensim is an implementation of topic modeling written in Python [18]. It includes implementations of LDA, LSI and DTM. For sentiment analysis, the API wraps Vader (https://github.com/cjhutto/vaderSentiment), a rule based model for sentiment analysis that uses a lexicon of English words [11]. As the preprocessor component, the processor interacts for storage with MongoDB, allowing to record each view of the data (e.g. a topic model view) as a document view of each opinion, with the attributes leveraged by the respective view. For instance, from the sentiment view of an opinion, each document in MongoDB stores neutral, positive and negative scores at sentence level. Weights for topic membership are stored in the topic model view of each opinion. Then, the documents stored in MongoDB ingest the indexer, the component that provides data for opinion search and browsing.

Indexer. This component is in charge of opinion indexing. For each view of the data, we create an index allowing search and browsing at different levels of granularity. As opinions are clustered using topic models, browsing is conducted using topics as opinion aggregation containers. For each topic, each opinion records its membership score, which indicates the degree of membership of each opinion to the topic. As each topic is a probability distribution over words, we store the weights of each word per topic. As browsing is conducted over topics, the use of words to describe each topic is a key element of Viscovery. To integrate the sentiment view of the data, we index opinions and their related sentiment weights for search and browsing. To ingest these indexes, we recover the document views created by the processor in the previous step, processing and indexing them into Elasticsearch (https://www.elastic.co/). Elasticsearch is a distributed, RESTful search engine capable of support searches over unstructured data implementing fast and efficient data access operations using inverted indexes. We use Elasticsearch indexes to support all the search and browsing operations in Viscovery.

DFR browser To visualize opinion trends we started using DFR browser (https://agoldst.github.io/dfrbrowser/), a visualization tool that works over topic models to integrate data views into a single, coherent, and searchable visualization of the data. As the code of DFR browser is available, we started working over DFR browser to cast this tool to our needs and requirements. DFR allows to

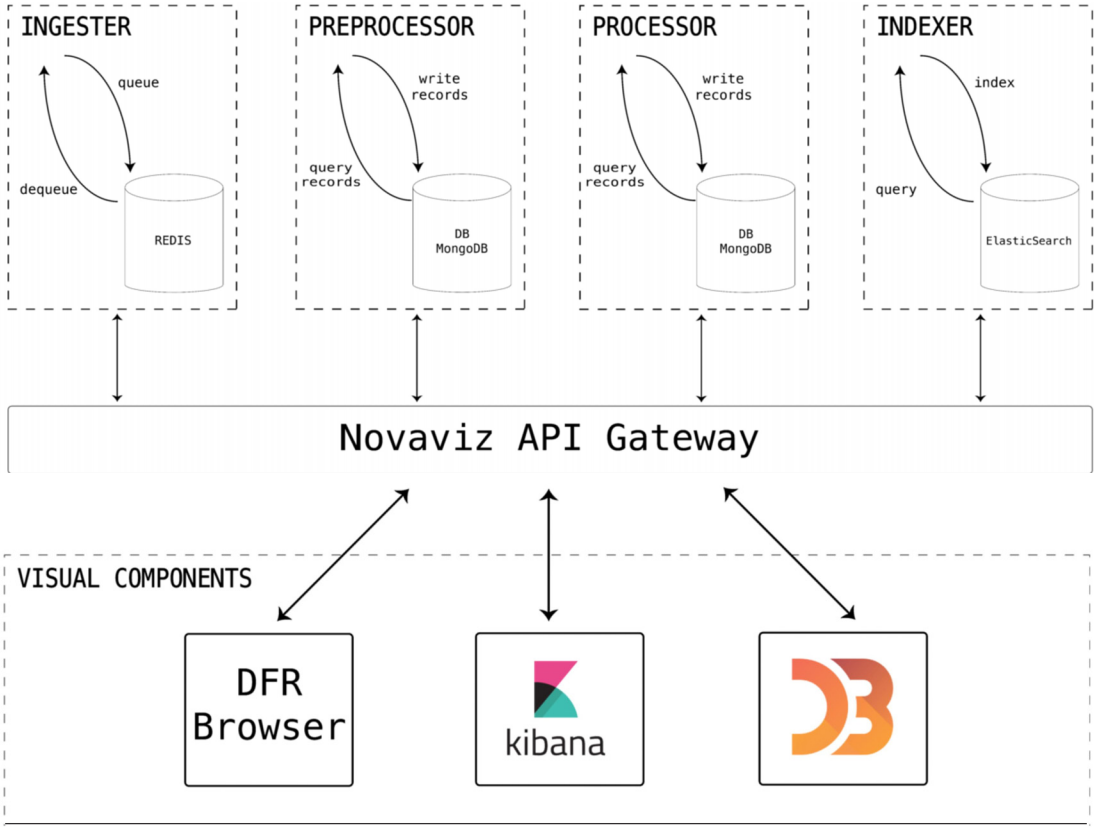


Figure 1. Viscovery architectural diagram. Data visualization components are connected to data processing components using our Novaviz API

search over topics, the basic search-able element in the visualization and to disaggregate the information at topic level into documents and words by topic.

Kibana Kibana is part of the suite provided by Elastic, named The Open Source Elastic Stack. The purpose of Kibana is to hand Elaticsearch visualizations.

D3.js Another tool that we use for data visualization is D3.js (https://d3js.org/). D3.js is a JavaScript library for data visualizations compatible with HTML, SVG, and CSS. D3 follows a data-driven approach for data manipulation using DOM as a standard for document representation.

5.2 DFR Cast

DFR (Data For Research) (http://agoldst.github.io/dfrbrowser/demo) is a visualization tool that produces global data views avoiding unnecessary accesses to documents at fine granularity levels. It produces a global first data view comprising document contents using topics as containers and words as topic descriptors. Topics can be picked in the global data view showing the most relevant words of the selected topic as a list. The temporal evolution of the topic is showed in the topic view. Lengthwise selection on the timeline of the topic exhibits data responsiveness, updating top-documents at topic level. Along with top document at topic level, a list of top-50 words is showed.

The document list includes three attributes per document: the title, the degree of membership of the document to the selected document, and the number of tokens that compound it. These lists include the top-20 most salient documents per topic in terms of degree of membership. The topic view provided by DFR is shown in Figure 2.

We extended the DFR topic view to include sentiment analysis and a 2D topic global view between the timeline and the list of top documents. To en-chase the topic embedding we modified the DRF topic view. By default, DFR does not include sentiment visualizations. Then two files, sentiment scores at term and document levels were included to allow sentiment visualizations. These files were used to indicate the polarity of topics, documents, and words. At topic level, each topic was colored according to its polarity, using a white/red color palette (negative scores were represented in red). In addition we included a button in the top words list to change the length of each word bar according to objective/ subjective scores. A screen shoot of our sentiment DFR topic-view extension is shown in Figure 3.

6. Data Slices and Results

6.1 Incremental Learning Testing

We evaluate our incremental version of DTM to measure speed up and model quality in terms of topic coherence [20]. Topic coherence has shown to be a very effective measure to evaluate the quality of a topic model [1]. Topic coherence corresponds to the correlation between the top-

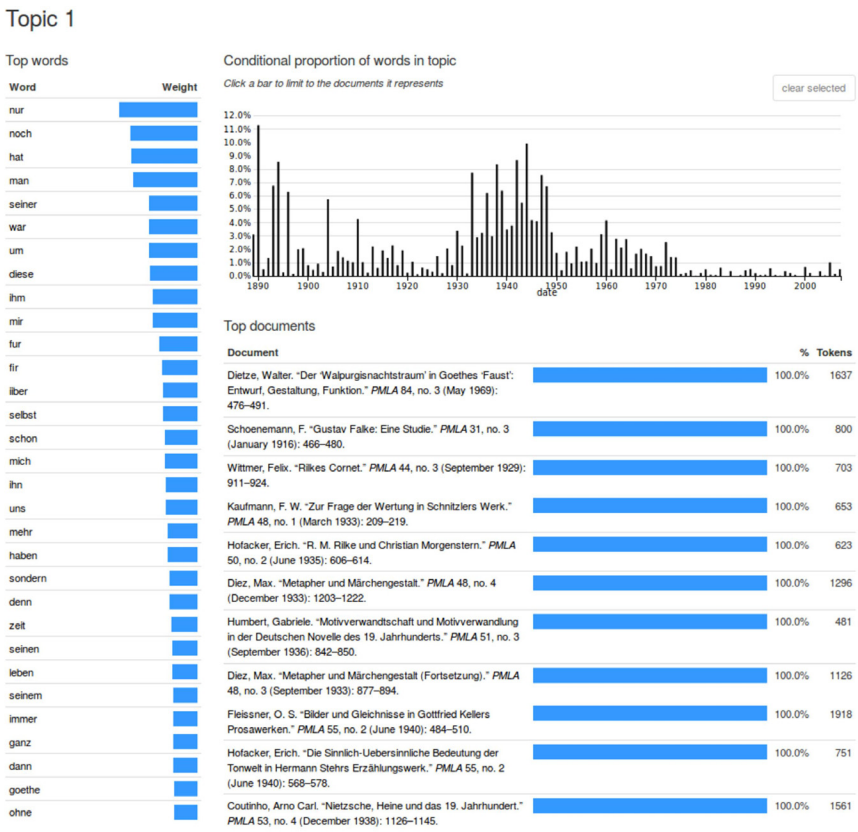


Figure 2. A generic DFR topic-view

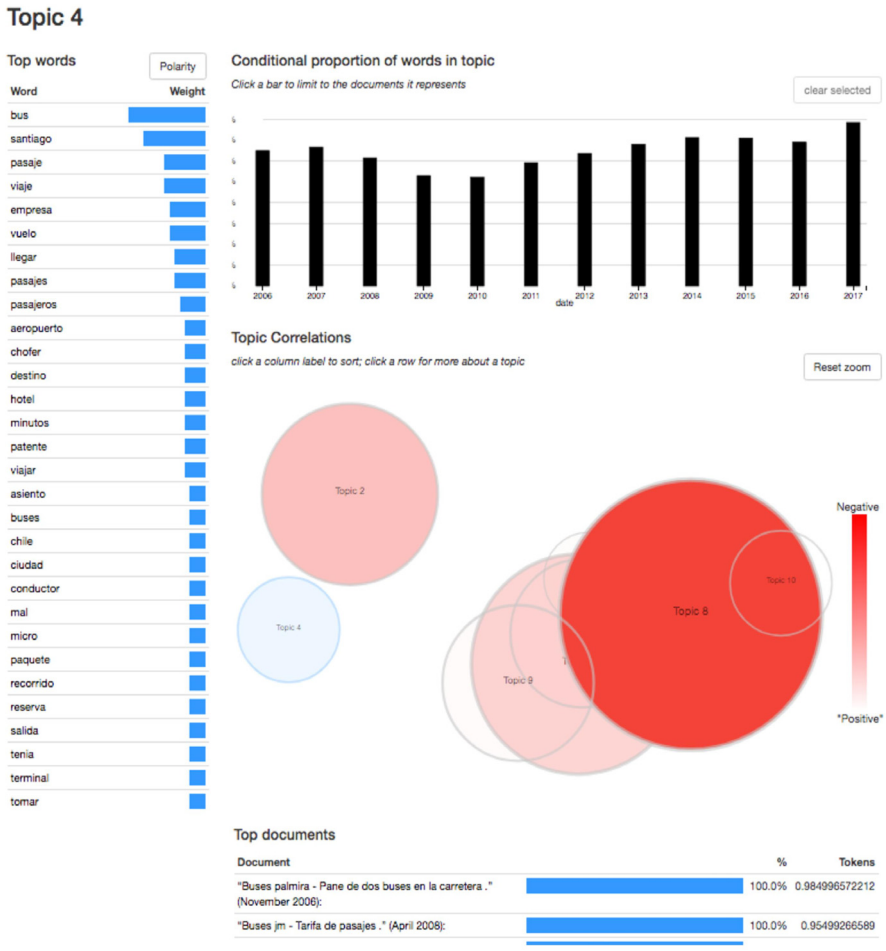


Figure 3. Our sentiment DFR topic-view

Algorithm	Mean Coherence	Variance Coherence	Average Computational Time
DTM	-1.5747	0.0047	2:06:56
DTM + Seq. Upd.	-1.5373	0.0063	1:47:37 + 0:11:50

Table 1. Topic coherence and computational times in DTM and DTM with sequential updating in the European Parliament corpus. Mean and variance over ten runs per algorithm are reported

terms of each topic according to their occurrences in a reference corpus. A high coherence indicates that the topics inferred from the model includes highly correlated terms.

We expect to reduce the computational time involved in model fitting avoiding a retrogress in terms of topic coherence. To test this aspect of our proposal, we run ten trials of model fitting for a corpus, with and without incremental learning on the last time slice. We used as test data a curated dataset to evaluate topic coherence provided by Greene & Cross [7] that comprises news about the political agenda of the European Parliament. The dataset is divided into four time slices and is compounded by 1324 news articles classified into a manually-specified number of topics, helping to evaluate topic coherence. Mean and variance of coherence and average computational times involved in both algorithms are reported in Table 1.

As expected, the mean computational time involved in DTM + Sequential Updating is less than the time registered by DTM, with only 11 minutes spent in the fourth slice. This result indicates that the most expensive step of the algorithm is the Kalman variational inference. As our proposal constraints the inference step two the last slices of the data, it reduces the cost involved. As the data used for this experiment is small (we used this dataset to estimate topic coherence) the difference between both algorithms in terms computational time is small.

In Reclamos.cl, a big data set with more than 200,000 complains that we indexed in Viscovery, the difference between both algorithms in terms of computational time is high. If we use DTM over the whole dataset, model fitting takes 14.7 hours. On the other hand, using sequential updating over the last time slice it takes 1.91 hours. Note that without sequential updating, we need to retrain the whole model for each new slice. Our proposal avoids this with a speed up of almost 8x.

6.2 Data Slices Over Reclamos.cl

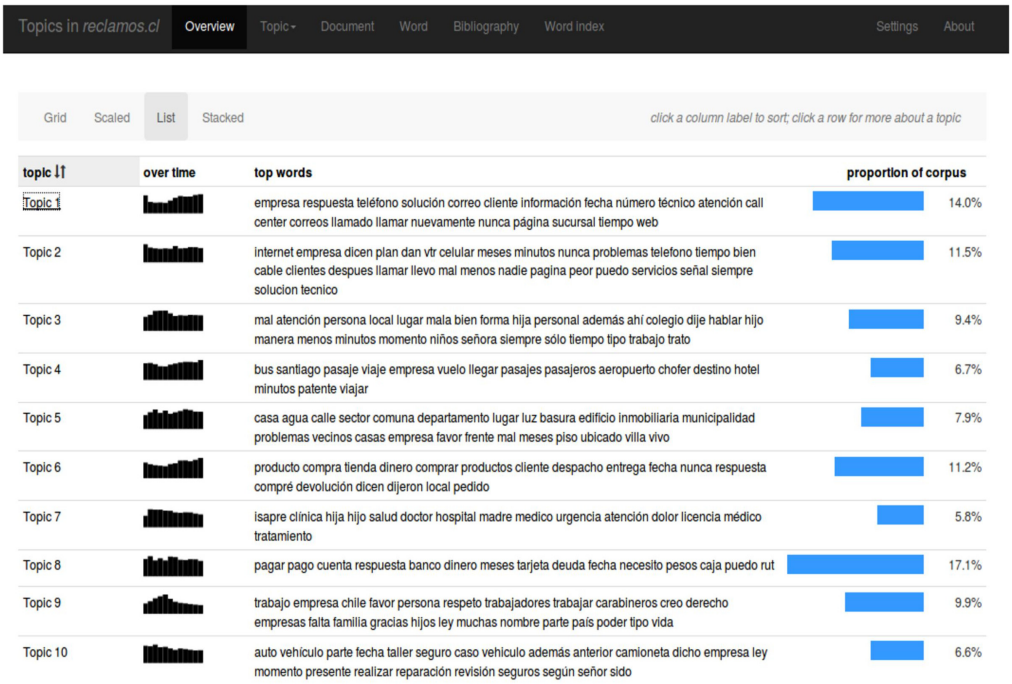
The current version of Viscovery implements browsable sentiment analysis at topic and word levels. Currently we are working on the implementation of sentiment analysis at document level, according to the proposal introduced in Section 4. We indexed into Viscovery 12 years of data from Reclamos.cl, a Chilean forum for complaints about companies, brands, and institutions. Reclamos.cl is a very active site in Chile, reporting an overall of 90,128 persons contacted by companies after complaint publication (a significant number in proportion to the Chilean population).

The dataset contains 201,969 different complaints. Retail, government, banks and universities are among the most frequent subjects of opinions. The size of the vocabulary after stop-word removal is 86,723 terms. We used 18 hours to create the model of reclamos using Viscovery.

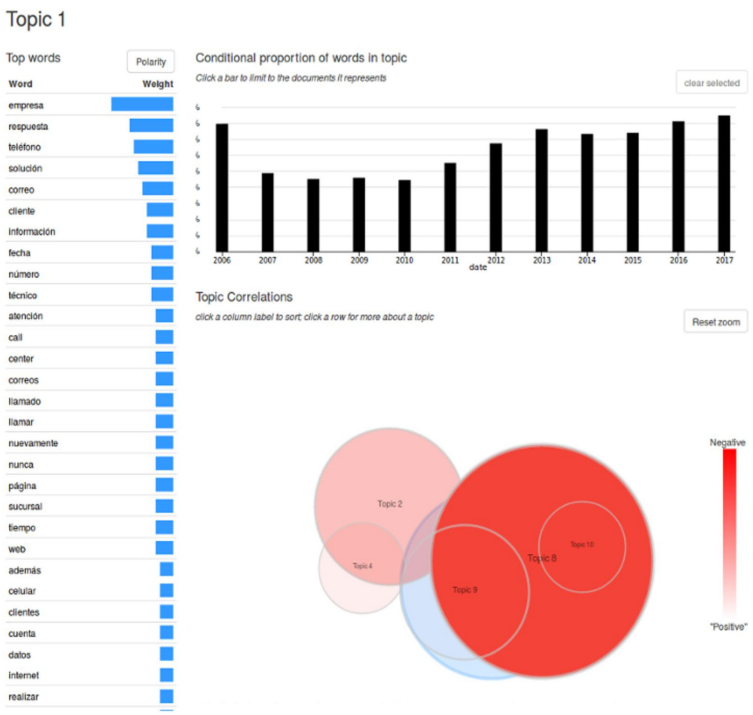
Running DTM using yearly-based time-stamps, we achieved 12 time slices of the data. We used the default value for the number of topics, set as 10 for this example. Data slices for this corpus are shown in Figure 4. As Figure 4 shows, the a) Corpus view (overview) has four alternatives for corpus deployment: grid, scaled, list, and stacked. We show topics using lists as data views. The list of topics include topic proportions over time, top words per topic and the proportion of the topic in the corpus. When a topic is selected (we pick topic 1 for this example), the b) Topic view is deployed. A topic view shows the list of top words for the topic, sorted in decreasing order according to the prominence of the word on the topic. If the polarity box is pressed the bars are modified according to the sentiment score of the word in the topic. For this version of Viscovery, the bar size is proportional to the sum of positive and negative scores. Currently we are implementing an extension that produces two bars per term, one per polarity. Topic embeddings are also shown in this view, illustrating the correlation between topics (distances in the topic embedding). The polarity orientation of each topic is shown using a color bar, where negative-biased topics are depicted with red shades. A list of top-documents per topic is shown below the embedding (omitted in this figure) and the user can select a specific opinion. In this case, Viscovery deploys the c) Document view, where the top words of the document in the given topic are shown. The subject and the date of the complaint is shown at the top of the view. Finally, Viscovery provides a d) Term view across topics, showing how relevant is a given word across topics. In the example, we show the word view using the term 'company'. As this view shows, this word is used in many topics of reclamos.cl, with different levels of membership. The topics are sorted in decreasing order according to the prominence of the word in each topic.

7. Conclusions

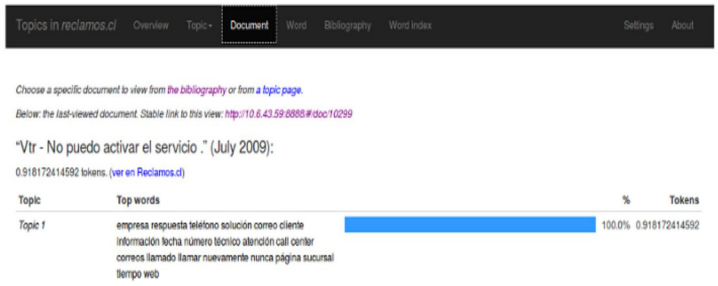
We present Viscovery, a tool for opinion browsing and



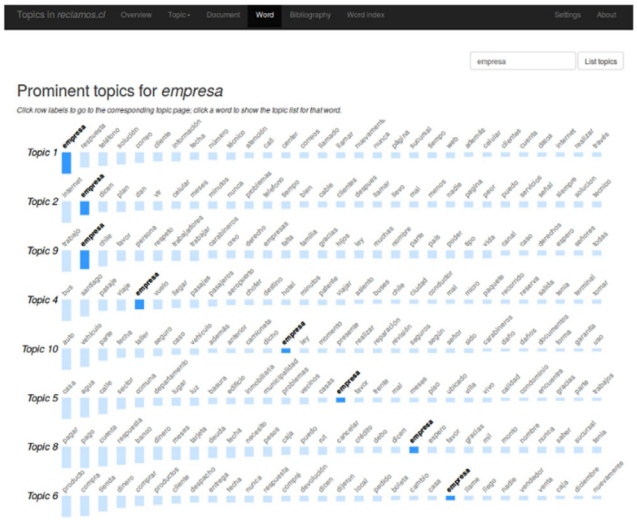
a) Corpus view across topics



b) Topic view with temporal proportions



c) Document view



d) Word view across topics

Figure 4. Viscovery data views. Three data slices are deployed from the (a) Corpus view after topic selection: (b) Topic view, which includes top words per topic, topic proportions on time, (c) Document view, depicting the membership of the document to the given topic, and (d) Word view across topics, showing the ranking of the word in each topic where the word is prominent

trend tracking. Key elements of Viscovery are Dynamic Topic Models (DTM) and our extension of DTM for sequential updating. We include sentiment analysis in Viscovery starting from sentiment scores at sentence level and then conducting aggregation across topics and documents. This approach is simple and effective. For visualization we use DFR browser, extending DFR to include topic embeddings and sentiment analysis.

Currently we are extending Viscovery to include more functionalities. Among these functionalities we are working on the sentiment-document view, topic evolution tracking view and opinion search module. We are implementing these modules using Kibana and D3, two visual components considered in Viscovery not included in the current version. In addition, we are using Viscovery to index more sources, as opinions retrieved from Twitter and Reddit.

Acknowledgment

This work was supported by the Fondef VIU 15E0085 project of the National Agency of Science and Technology, Conicyt, Chile. Dr. Mendoza was partially funded by the project BASAL FB0821. The funder played no role in the design of this study.

References

[1] Blair, S., Bi, Y., Mulvenna, M. (2016). Increasing topic coherence by aggregating topic models. *In:* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), V. 9983 LNAI, p. 69–81.

[2] Blei, D., Lafferty, J. (2006). Dynamic topic models. *In:* Proceedings of the 23rd International Conference on *Machine Learning, ICML*, p. 113–120.

[3] Blei, D., Ng, A., Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3 (4-5) 993–1022.

[4] Bravo-Marquez, F., Mendoza, M., Poblete, B. (2014). Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, 69 (1) 86–99.

[5] Dempster, A., Laird, N., Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistics Society*, 39.1–38.

[6] Eguchi, K., Lavrenko, V. (2006). Sentiment retrieval using generative models. *In*: *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing, EMNLP*, p. 345–354.

[7] Greene, D., Cross, J. P. (2016). Exploring the political agenda of the european parliament using a dynamic topic modeling approach. *CoRR*, abs/1607.03055.

[8] Griffiths, T., Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences, PNAS*, 101 (1) 5228–5235.

[9] Harvey, M., Ruthven, I., Carman, M. (2010). Ranking social bookmarks using topic models. *In*: Proceedings of the 19th International Conference on Information and Knowledge Management, CIKM, p. 1401–1404.

[10] Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42 (2) 177–196.

[11] Hutto, C., Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. *In*: Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM.

[12] Jo, Y., Oh, A. (2011). Aspect and sentiment

unification model for online review analysis. *In*: *Proceedings of the 4th ACM International Conference on Web Search and Data Mining, WSDM*, p 815– 824.

[13] Kurland, O., Lee, L. (2009). Clusters, language models, and ad hoc information retrieval. *ACM Transactions on Information Systems, TOIS*, 27(3).

[14] Lin, C., He, Y. (2009). Joint sentiment/topic model for sentiment analysis. *In*: Proceedings of the 18th ACM International Conference on Information and Knowledge Management, CIKM, p 375–384.

[15] Mei, Q., Ling, X., Wondra, M., Su, H., Zhai, C. (2007). Topic sentiment mixture: modeling facets and opinions in weblogs. *In*: Proceedings of the 16th ACM International World Wide Web Conference, WWW, p 171–180.

[16] Mendoza, M., Ormeno, P., Valle, C. (2018). Boosting text clustering using topic selection. *In*: Proceedings of the International Conference on Pattern Recognition Systems, ICPRS, IET Conference Publications, number CP745, p, 46–51.

[17] Newman, D., Hagedorn, K., Chemudugunta, C., Smyth, P. (2007). Subject metadata enrichment using statistical topic models. *In*: Proceedings of the ACM International Conference on Digital Libraries, ICDL, p. 366–375.

[18] XehoYek, R., Sojka, P. (2010). Software Framework for Topic Modeling with Large Corpora. *In*: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, p. 45–50.

[19] Sie, S.-H. and Yeh, J.-H. (2014). Library book recommendations based on latent topic aggregation. *In:* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), V 8839, p. 411–416.

[20] Wallach, H., Murray, I., Salakhutdinov, R., Mimno, D. (2009). Evaluation methods for topic models. *In*: Proceedings of the 26th International Conference on Machine Learning, ICML, p. 1105– 1112.

[21] Yamada, T. (2017). Detection of topics from newspaper and its analysis of temporal variations in regions. *In*: Proceedings of the 2017 Pacific Neighborhood Consortium Annual Conference and Joint Meetings: Data Informed Society, PNC, p. 44–49.

Appendix. Lower Bound of the Likelihood for the Incremental Algorithm

In this section we give details of the incremental algorithm that maximizes the lower bound of the likelihood on $\log p(d_{1:T})$. This section is an extension of the appendix provided in Blei and Lafferty [2] where the lower bound is calculated for the static algorithm. For the incremental version of the inference algorithm, we only need to calculate the terms for the last time slice T. The first term of the lower bound is:

$$\begin{aligned}\mathbf{E}_q \log_p(\beta_T|\beta_{T-1}) &= -\tfrac{V}{2}(\log\sigma^2 + \log(2\sigma^2) \\ &\quad -\tfrac{1}{2\sigma^2}\mathbf{E}_q(\beta_T-\beta_{T-1})^T(\beta_T-\beta_{T-1}) \\ &= -\tfrac{V}{2}(\log\sigma^2+\log 2\pi) - \tfrac{1}{2\sigma}||\tilde{m_T} - \tilde{m_{T-1}}||^2 \\ &\quad -\tfrac{1}{\sigma^2}Tr(\tilde{V_T}) + \tfrac{1}{2\sigma^2}(Tr(\tilde{V_0}) - Tr(\tilde{V_T})) \qquad (16)\end{aligned}$$

The second term is:

$$\begin{aligned}&\mathbf{E}_q \log_p(d_T|\beta_T) = \sum_w n_{tw}\mathbf{E}_q(\beta_w - \log\textstyle\sum_w \exp(\beta_w)) \\ &\geq \textstyle\sum_w n_w\tilde{m}_w - n_w\zeta_T^{-1}\sum_w \exp(\tilde{m_T} + \tilde{V}_w/2) \\ &+n_T - n_T\log\zeta_T^{-1} \qquad (17)\end{aligned}$$

where $n_T = \sum_w n_w$. The third term is the entropy $H(q) = \frac{1}{2}\sum_w \log\tilde{V}_w + \frac{V}{2}\log 2\pi$. The term $\frac{V}{2}\log 2\pi$ is canceled in term 1 and the entropy. In term 2:

$$n_T\zeta_T^{-1}\textstyle\sum_w \exp(\tilde{m_T} + \tilde{V}_w/2) = n_T\zeta_T^{-1}\zeta_T = n_T \quad (18)$$

The new term -n_T is canceled with the corresponding n_T. Then, the bound can be obtained as:

$$\begin{aligned}&= -\tfrac{V}{2}(\log\sigma^2) - \tfrac{1}{2\sigma}||\tilde{m_T} - \tilde{m_{T-1}}||^2 - \tfrac{1}{\sigma^2}Tr(\tilde{V_T}) \\ &+\tfrac{1}{2\sigma^2}(Tr(\tilde{V_0}) - Tr(\tilde{V_T})) + \textstyle\sum_w n_w\tilde{m}_w - n_T\log\tilde{\zeta_T} \\ &+\tfrac{1}{2}\textstyle\sum_w \log\tilde{V}_w \qquad (19)\end{aligned}$$