分散ストリーム処理基盤Stormと言語モデリングによる新情報を含む文書の検出

川原　駿† 関　和広†† 上原邦昭 †††

†, ††† 神戸大学大学院システム情報学研究科　〒657–8501 兵庫県神戸市灘区六甲台町 1-1
†† 甲南大学知能情報学部　〒658–8501 兵庫県神戸市東灘区 8-9-1
E-mail: †kawahara@ai.cs.kobe-u.ac.jp, ††seki@konan-u.ac.jp, †††uehara@kobe-u.ac.jp

あらまし　Wikipedia などの知識ベースは複数の編集者によって記事の編集が行われている．しかし，オンラインニュースやソーシャルメディアなど近年の情報爆発により，適切な編集・更新が追い付いていないのが現状である．そこで本論文では，ウェブ上の膨大な情報を自動的にフィルタリングし，知識ベースの記事の更新が必要となるような新情報を実時間で検出するシステムを提案する．新情報の検出には，言語モデリングを用いた 2 段階のフィルタを構築する．このフィルタを分散ストリーム処理基盤 Storm 上に実装することにより，大規模ストリームデータのフィルタリングを可能とする．TREC KBA Stream Corpus を用いた実験により，提案システムの有効性を示す．

キーワード　不適合フィードバック，リアルタイム，並列分散，TREC，KBA，Wikipedia

1. はじめに

Wikipedia や Freebase に代表されるような知識ベース（Knowledge base）には膨大な数の記事が蓄積されており，多くの人々に利用されている．実際に Wikipedia の英語版には 450 万件以上の記事が存在する．これらの記事は少数の人間により管理されており，英語版 Wikipedia の記事の編集者（管理者）はおよそ 1300 人存在している (注1)．単純に 450 万件の記事を 1300 人で管理するなら，1 人当たり 3500 件近い記事を担当しなければならない．一方で，近年は Twitter などの SNS やブログサービスを利用して誰でも気軽に情報発信を行えるようになっており，ウェブ上ではこの瞬間にも新たな情報が次々と発信されている．Wikipedia の記事の編集者は，これらの情報を逐一確認して記事に反映させなければならないものの，発信される情報が多すぎるため，現実には多くの記事において情報が反映されるまでに 1 年もの時間を要することも少なくない [7].

そこで本研究では，ウェブ上の情報を自動的にフィルタリングし，知識ベースの記事の更新が必要となるような新情報を実時間で検出するシステムを提案する．以降，知識ベースの記事の見出しとなっている人物・施設・組織や概念を「エンティティ（entity）」と呼ぶ．例えば,「バラク・オバマ」「ホワイトハウス」「民主党」などは全てエンティティである．本研究で提案するシステムは，エンティティに関連する新情報を検出することを目的とする．このようなシステムの開発は，情報検索の国際ワークショップである Text REtreival Conference（TREC）の Knowledge Base Acceleration（KBA）トラックの Vital Filtering タスク [6] でも課題とされており，盛んに研究が行われている．しかし，このタスクの参加者によって提案されたシステムには 3 つの問題がある．すなわち，大規模なストリームデータをどのように扱うかが考慮されていないこと，分類モデルの学習に要するデータが少ない点を考慮できていないこと，関連する文書の中から記事の更新が必要となるような新情報をうまく区別できていないことである．本研究で提案するシステムは，分散リアルタイム処理システム Apache Storm (注2) 上に構築し，大規模なストリームデータをリアルタイムに処理することを可能としている．また，学習データの不足に関しては非関連文書のデータを用いて不適合フィードバックを行うことにより対処し，記事の更新が必要となるような情報を表す言語モデルを構築してその区別を行う．

本論文の構成は以下の通りである．まず，2 章で過去に Vital Filtering タスクで提案されたシステムを説明し，その問題点を明らかにする．3 章では，Apache Storm の概要と提案システムの構造を述べ，4 章では TREC KBA Stream Corpus を用いて実際にフィルタリング実験を行い，その結果を考察する．最後に，5 章で本論文のまとめと今後の課題を述べる．

2. 関連研究

過去の Vital Filtering タスクにおいて，多くの新情報検出システムが提案されている．Liu ら [10] や Dietz と Dalton [5] などは，ターゲットとしているトピックに関連するトピックの情報を利用することで，分類に必要な情報を拡張する手法を提案している．Abbes ら [1] を始めとしたいくつかの研究チームは，トピックに関するキーワードの文書中での出現位置や，過去数時間以内にトピックについて言及された文書の数を素性として利用している．Kenter [8] を始めとしたいくつかの研究チームは，コサイン類似度やジャカード類似度などの言語情報を利用した文書とトピックの類似度を用いている．また，Wang

(注1)：http://ja.wikipedia.org/wiki/Wikipedia:全言語版の統計 参照

(注2)：http://storm.apache.org/

ら [12] や Bellogin ら [3] は上記に挙げた手法や素性を全て利用して分類器を構築している．ただし，Bellogin らを始めとした多くの研究チームはトピック毎に分類器を構築しているのに対し，Wang らは全てのトピックで共通の分類器を 1 つ構築することで学習データの不足に対応し，最もよい性能を示した．しかし，この手法ではトピックに特有の特徴を捉えることができないことが問題となる．本研究では，非関連文書のデータも利用することにより学習データを確保し，トピック毎の言語モデルを構築することで，この問題に対処する．

上記に挙げた素性のほとんどは記事の更新が必要となるような新情報をうまく区別できないことも問題である．例えば，bag-of-words や TFIDF などを素性として文書やトピックをベクトル化し，その類似度を計算するだけでは，トピックに関連する文書かどうかを判断するのには有効だとしても，その文書が新情報を含むかどうかの判断には有効ではない [2]．そこで本研究では，2 つの言語モデルを構築し，そのいずれかを使用する．1 つは，知識ベースの記事を言語モデル化したものである．そして，このモデルとの類似度が低い文書に新情報が含まれていると判断する．これは，知識ベースに掲載されている内容と類似した文書は新情報を含む可能性が低いと考えられるからである．もう 1 つは，過去に新情報を含むと判断された文書における頻出語から構築した言語モデルである．新情報を含む文書に共通して現れる単語を含むような文書もまた，新情報を含む可能性が高いと考え，このモデルとの類似度が高い文書に新情報が含まれていると判断する．

また，上記に挙げたシステムはいずれも際限なく発信されてくる文書，即ちストリームデータの扱いについて考慮されていない．つまり，仮にコーパスによる実験で高い検出能力を示したとしても，大規模なストリームデータに対応できるとは限らない．本研究では，分散リアルタイム処理システム Apache Storm を導入し，大規模なストリームデータの処理が可能なシステムを提案する．

3. 提案システム

本研究で提案するシステムは，分散リアルタイム処理システムである Apache Storm 上に，2 段階のフィルタリングシステムを構築する．そこで以下では，まず Apache Storm について説明してから，フィルタリングシステムの概要，詳細を述べる．

3.1 Apache Storm

Apache Storm とは，分散リアルタイム処理システムであり，際限のないデータストリームの処理を可能とする．ユーザは図 1 に示すような「トポロジー」（topology）を定義し，Storm クラスタ上に起動するだけでストリーム処理を実装することができる．トポロジーとは処理の流れを表したグラフ構造のことである．トポロジーの各節点には処理を定義し，各節点間の枝 (注3) によりデータの流れを定義する．

節点は 2 種類存在し，それぞれ「スパウト」（spout）,「ボルト」（bolt）と呼ぶ．スパウトはストリームの発生源であり，データストリーム上のデータを Storm で扱える「タプル」（tuple）という形式のデータに変換する．本システムの場合は，処理する文書 1 件 1 件をタプルに変換する．ボルトはストリームからタプルを受け取り，定義された処理をタプルに対して実行する．そして，必要があれば新たなストリームにタプルを送り出す．本システムの場合は，ストリームから文書タプルを受け取り，その文書が関連文書であれば新たなストリームに送り出すボルトを構築する．また，各ボルトは並列に実行され，並列数はユーザが指定することができる．

(注3)：枝のことを「ストリーム」（Stream）と呼ぶ．

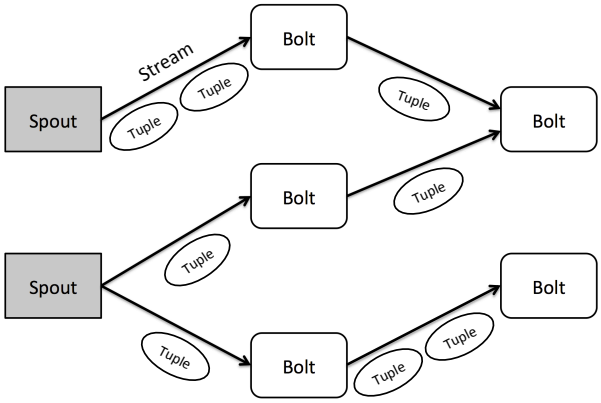

図 1　Storm のトポロジー．

3.2 システムの概要

図 2 は提案システムのトポロジーを示したものである．入力スパウト（input spout）はデータストリーム上の文書を文書タプルに変換し，表記名フィルタ（sfn filter bolt）ではエンティティの表記名（surface form name；SFN）を含む文書かどうかを判定する．そして，関連文書フィルタ（relevant filter bolt）では関連文書かどうかを判定し，重要文書フィルタ（vital filter bolt）では文書が新情報を含むかどうかを判定する．このように，フィルタリング処理は 2 段階に分けて行う．各ボルトは文書タプルを並列に処理する．以下，各ノードについてもう少し詳しく説明していく．

入力スパウトはデータストリーム上の個々の文書を文書タプルに変換し，表記名フィルタに送り出す．なお，各文書は予め 3.3 節で説明する前処理が行われている．

表記名フィルタは文書タプルがエンティティの表記名を含むかどうかをチェックし，表記名を含む文書タプルのみを関連文書フィルタに送り出す．エンティティの表記名は 1 つのエンティティにつき複数存在する．例えば,「バラク・オバマ」というエンティティの表記名は「バラク・オバマ」「バラク・フセイン・オバマ」「バラク・H・オバマ」などがある．これらの表記名のうち少なくとも 1 つでも含む文書タプルのみを関連文書フィルタに送り出す．

関連文書フィルタは文書タプルが関連文書かどうかを判定し，関連文書と判定された文書タプルのみを重要文書フィルタに送り出す．関連文書かどうかの判定には，非関連文書モデルを利用する．詳細は 3.4 節で述べる．

重要文書フィルタは，関連文書フィルタにより関連文書と判定された文書が新情報を含むかどうかを判定し，その判定結果

とともに出力ボルト（output bolt）に送り出す．なお，以後文書が知識ベースの記事の更新が必要となるような新情報を含む場合を「重要」（vital）であると呼ぶことにする．一方，関連文書ではあるが「重要」ではない文書は「準重要」（useful）であると呼ぶことにする(注4)．さらに,「重要」と判定された文書はモデル更新器（wikipedia update bolt）（詳細は 3.6 節で述べる）にも送り出す.「重要」であるかどうかの判定には知識ベースモデルまたは重要文書モデルのいずれかを利用する．詳細は 3.5 節で述べる．

最後に，並列に処理された重要文書フィルタのフィルタリング結果を出力スパウトで受け取って出力する．

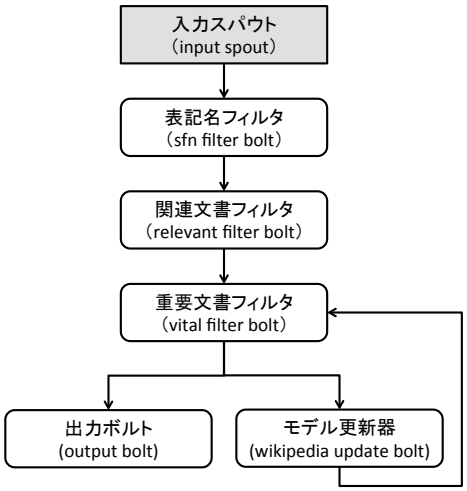

図 2　システムのトポロジー.

3.3　文書の前処理

本システムで用いる文書には予め以下の処理を施している．

- 文書中の単語の小文字化.
- ストップワードと句読点の除去.
- 接辞処理 [11].
- 文書の言語モデル化.

文書の言語モデル化において文書 d に対する言語モデル $p(w|d)$ は，ディリクレスムージング [14] に基づき式 (1) で表される．

$$p(w|d) = \frac{c(w,d) + \mu p(w|\mathcal{C})}{|d| + \mu} \tag{1}$$

バックグラウンド言語モデル $p(w|\mathcal{C})$ には Google-Ngram (注5)（ユニグラム）を使用した．また，μ の値は [14] で推奨されている値である 2000 を用いる．

3.4　関連文書フィルタ

関連文書フィルタは文書タプルが関連文書かどうかを判定し，関連文書と判定された文書タプルのみを重要文書フィルタに送り出す．関連文書かどうかの判定には，MultiNeg [13] によって構築した非関連文書モデル（Negative Language Model, NLM）を利用する．

MultiNeg はアドホック検索の検索結果を，非関連文書を利用した不適合フィードバックにより改善するために提案されたモデルであり，検索結果の適合度が低いクエリに対する検索に有効であることが示されている．MultiNeg では，非関連文書モデルと検索対象の文書の言語モデルの類似度により，文書の関連度を調節する．なお，ここで言う非関連文書とは，初期の検索結果のうちクエリに非関連であった文書のことを指す．例えば，ユーザがアップル社に関する情報を検索したい場合を考える．クエリとして「アップル」を利用し，もしりんごに関する文書が検索結果に含まれていたら，これらの文書は非関連文書である．

MultiNeg では非関連文書集合 $L = \{l_1, ..., l_f\}$ から非関連文書モデル $\Theta = \{\theta_1, ..., \theta_f\}$ を構築する．モデルの推定には EM アルゴリズム [4] を利用する．MultiNeg を利用する場合は，クエリとの類似度計算にカルバック・ライブラー情報量検索モデル [9] を利用する．カルバック・ライブラー情報量検索モデルに基づく類似度は式 (2) のようにクエリ q の文書 d に対する負のカルバック・ライブラー情報量で表される．

$$S(q,d) = -D(\theta_q||\theta_d) = -\sum_{w \in V} p(w|\theta_q) \log \frac{p(w|\theta_q)}{p(w|\theta_d)} \tag{2}$$

カルバック・ライブラー情報量検索モデルにより求めた類似度に，非関連文書モデルに基づくペナルティ $S(NLM, d)$ を与え，MultiNeg ではこの修正後の値（式 (3)）を類似度とする．

$$S(q,d) - S(NLM,d) \tag{3}$$

ペナルティは，f 個の非関連文書モデルと文書 d との類似度をそれぞれ計算し，その最大値，即ち，

$$\begin{aligned} S(NLM,d) &= \max(\bigcup_{i=1}^{f} \{S(\theta_i, \theta_d)\}) \\ &= -\min(\bigcup_{i=1}^{f} \{D(\theta_i||\theta_d)\}) \end{aligned} \tag{4}$$

で与えられる．

本システムにおいては，式 (4) の値をエンティティと文書の類似度とみなして，予め決められた閾値 t_r より小さければ関連文書と判定する．また，非関連文書モデルの構築には後述する訓練データを利用する．

3.5　重要文書フィルタ

重要文書フィルタは，関連文書フィルタにより関連文書と判定された文書が新情報を含むかどうか，すなわち「重要」であるかどうかを判定する.「重要」であるかどうかの判定には知識ベースモデル（Knowledge base Article Language Model，KALM）または重要文書モデル（Vital Language Model，VLM）のいずれかを利用する．

知識ベースモデルはエンティティの知識ベースの記事を言語モデル化したものである．実際の Wikipedia の記事のテキストと，既に「重要」であると判定された文書（訓練データ）からユニグラム言語モデルを構築する．

(注4)：「重要」である文書と「準重要」である文書をまとめたものが「関連文書」となる.

(注5)：http://googleresearch.blogspot.jp/2006/08/all-our-n-gram-are-belong-to-you.html

一方，重要文書モデルは「重要」な文書における頻出語から構築したユニグラム言語モデルである．過去に検出された「重要」な文書群と「準重要」な文書群（訓練データ）を利用したカイ二乗検定により，両者を区別するのに有用と考えられる，「重要」な文書群固有の頻出語を抽出する．

フィルタリング時は，上記のモデルと文書の類似度を式 (2) で算出し，予め定めてある閾値と比較することにより,「重要」であるかどうかを判定する．知識ベースモデルを用いた場合は，その閾値 t_{vk} より小さければ「重要」であると判定する．これは，既に知識ベースに掲載されている内容を含む文書は「重要」ではない，すなわち，知識ベースモデルに類似した文書は「重要」ではないという仮定に基づく．一方，重要文書モデルを利用した場合は，その閾値 t_{vv} より大きければ「重要」であると判定する．こちらは,「重要」な文書群固有の頻出語から構築した重要文書モデルに類似した文書もまた「重要」であるという仮定に基づく．

3.6 モデル更新器

モデル更新器は重要文書フィルタで「重要」であると判定された文書を受け取り，知識ベースモデルや重要文書モデルを更新する．このモデル更新は「重要」な文書があれば編集者に推薦し，その文書に基づいて記事が更新されるという流れを反映したものである．

4. 評価実験

本論文では，2 つの評価実験を行った．1 つは TREC KBA の Vital Filtering タスクのルールに沿った評価実験，もう 1 つは，Storm を利用することによる処理速度の向上を示す実験である．

前者の Vital Filtering タスクでは，用意されたコーパス（データセット）を用いてフィルタリングを行い，その結果を F 値（F1-measure）で評価する．コーパスに含まれる文書にはタイムスタンプが付与されており，その順に読み込んでフィルタリングを行う必要がある (注6)．これは，Web 上の文書ストリームを擬似的に再現するためである．フィルタリング対象として 67 のエンティティが与えられる．

後者の実験では，フィルタリングの実施方法は Vital Filtering タスクと同様とし，その際の並列数（ボルト数）を変化させることにより，実行時間がどのように変化するかを検証する．並列数を変化させるのは関連文書フィルタのみとし，その他のボルトは並列化を行わない．これは，本システムにおけるボトルネックが関連文書フィルタだからである．

以下，Vital Filtering タスクで使用するコーパスや評価方法の詳細を述べた後に実験結果を示し，考察を述べる．

4.1 コーパス

Vital Filtering タスクではコーパスとして TREC KBA Stream Corpus 2014 (注7) が提供された．同コーパスは 2011 年 10 月 5 日〜2013 年 4 月 30 日までの 19ヶ月間，20,494,260 件の文書で構成されている．これらの文書はインターネット上のニュースやブログなど様々なものが存在し，言語は英語である．コーパスはエンティティ毎に訓練データとして利用できる訓練期間が定められている．その期間中の文書には正解ラベルが付与されており，訓練データとして自由に利用可能である．評価実験は，訓練期間に含まれない期間のデータを用いて行う．

本研究では効率的に実験を行うために，コーパスに対して予め以下のような前処理を施した．

- エンティティの表記名を 1 つも含まない文書を除外する．
- 重複した文書は 1 件を残して除外する．

4.2 評価方法

Vital Filtering タスクでは，フィルタリング結果を全てのエンティティのマクロ平均による F 値（macro-averaged F1-measure）により評価する．

$$\mathrm{F1}_{\mathrm{macro\_ave}} = \frac{2 \cdot \mathrm{P}_{\mathrm{ave}} \cdot \mathrm{R}_{\mathrm{ave}}}{\mathrm{P}_{\mathrm{ave}} + \mathrm{R}_{\mathrm{ave}}} \tag{5}$$

$$\mathrm{P}_{\mathrm{ave}} = \frac{1}{|E|} \sum_{e \in E} \mathrm{P}(e) \tag{6}$$

$$\mathrm{R}_{\mathrm{ave}} = \frac{1}{|E|} \sum_{e \in E} \mathrm{R}(e) \tag{7}$$

ここで，P，R はそれぞれ適合率と再現率を表す．E はフィルタリング対象の 67 のエンティティの集合である．

4.3 実験設定

フィルタリングに用いる各モデル（非関連文書モデル，知識ベースモデル，重要文書モデル）は訓練データを用いて構築した．非関連文書モデルは，訓練データ中の「エンティティの表記名を含むが関連ではない文書」を非関連文書とみなし，それらの文書を用いて構築した．ただし，上記の定義に当てはまるような文書が存在しないエンティティに関しては非関連文書モデルを構築しないこととした．この場合は関連文書フィルタは無条件で通過するものとした．知識ベースモデルは，Wikipedia の記事データとして 2012 年 1 月 4 日の Wikipedia のダンプファイル (注8) を利用し，また，訓練データ中の「重要」な文書も利用して構築した．ただし，エンティティによっては Wikipedia の記事が存在しない場合もあり，その場合は訓練データのみから構築した．

フィルタリングに用いる各閾値は次のように定めた．関連文書フィルタの閾値 t_r は訓練データ内の関連文書に対してフィルタリングを行い，それらが全て関連だと判定される閾値のうち，最小の値とした．重要文書フィルタの閾値 t_{vk}，t_{vv} は訓練データ内の関連文書に対してフィルタリングを行い，F 値が最も高くなるような閾値に設定した．

4.4 実験結果

本実験では，表 1 に示すような 5 つの異なる設定で実験を行った．Exact Match はエンティティの表記名を含む文書を全て「重要」とみなすシステムであり，図 2 における関連文書フィルタや重要文書フィルタによるフィルタリングは実行しな

(注6)：正確には 1 時間毎で構わない．

(注7)：http://s3.amazonaws.com/aws-publicdatasets/trec/kba/index.html20120104/index.html

(注8)：http://s3.amazonaws.com/aws-publicdatasets/trec/kba/enwiki-

い．このシステムを本実験のベースラインとした．KALM と VLM は重要文書フィルタにそれぞれ知識ベースモデル，重要文書モデルを使用したシステムである．なお,「更新無」が付与されているシステムは，モデル更新器を使用しない，すなわちモデルの更新を行わないことを表す．

本システムにより Vital Filtering タスクによる実験を行った結果を表 2 および表 3 に示す．Vital Filtering タスクには 2 つのサブタスクが存在し,「重要」な文書のみを正例とみなす「重要文書検出タスク」（表 2），関連文書を正例とみなす「関連文書検出タスク」（表 3）がある．表 2 より，重要文書検出タスクにおいては VLM が最も高い性能を示した．しかし，t 検定においてベースライン（Exact Match）との有意差は KALM のみ確認された（有意水準 1%で有意）．また，モデル更新を行うことにより，KALM も VLM も僅かではあるが性能の向上が見られた．関連文書検出タスクにおいては文書の関連・非関連のみが焦点となるため，重要文書フィルタによるフィルタリングはこの実験結果には影響しない．このため，KALM も VLM も同じ結果となっている．こちらでは，ベースラインからの F 値の向上が見られ，有意水準 1 %の t 検定において有意差が確認された．非関連文書を利用した関連文書のフィルタリングは有効であると言える．

Storm を利用することによる処理速度の向上を示す実験の結果を図 3 及び表 4 に示す．並列数の増加に伴って実行時間も短くなっていったが，並列数が 5 を超えた辺りで変化がほとんどなくなった．

表 1　実験を行った 5 つのシステム

システム	説明
Exact Match	表記名を含む文書を全て「重要」とみなす．
KALM	重要文書フィルタに知識ベースモデルを使用．
KALM（更新無）	〃（モデル更新を行わない）
VLM	重要文書フィルタに重要文書モデルを使用．
VLM（更新無）	〃（モデル更新を行わない）

表 2　Vital Filtering タスクによる実験の結果（重要文書検出タスク）．

System	Prec.	Recall	F_1	SU
Exact Match	0.106	0.993	0.192	0.062
KALM	0.120	0.876	0.211	0.077
KALM（更新無）	0.117	0.884	0.206	0.076
VLM	0.139	0.641	0.228	0.136
VLM（更新無）	0.153	0.428	0.225	0.172

表 3　Vital Filtering タスクによる実験の結果（関連文書検出タスク）．

System	Prec.	Recall	F_1	SU
Exact Match	0.284	0.995	0.442	0.258
KALM/VLM	0.313	0.978	0.474	0.294

表 4　関連文書フィルタの並列数と実行時間の関係．

並列数	実行時間（ms）
1	1,516,629
2	757,310
3	566,601
4	474,724
5	389,927
6	369,289
7	369,258
8	360,672
9	362,753

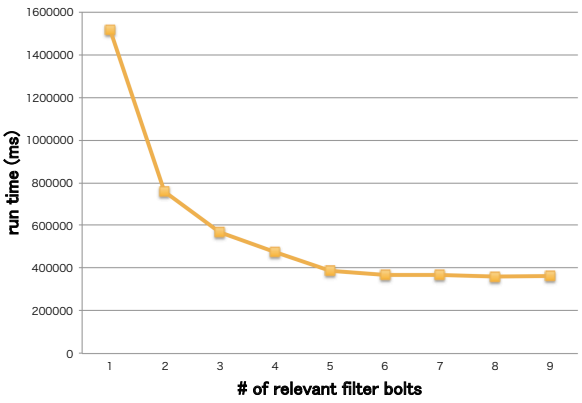

図 3　関連文書フィルタの並列数と実行時間の関係．

4.5 考　　察

Vital Filtering タスクの実験において VLM に有意差が見られなかったのは，VLM の結果がエンティティ毎に大きなバラつきがある（分散が大きい）ことに起因するものと考えられる．図 4 及び図 5 に，エンティティ毎に F 値がベースラインと比較してどのくらい向上したかを示す．VLM では，Ralph_Dannenberg や James_Windle で大きな向上が見られた一方で，Lizette_Graden や Corisa_Bell などでは大きく低下してしまっていた．一方 KALM ではほとんどのエンティティにおいて向上が見られた．

非関連文書モデルの構築時に，構築に利用した非関連文書の数がエンティティにより大きく異なっていた．非関連文書の数が性能に与える影響を調べるために，図 6 に示すような散布図を作成した．横軸に非関連文書の数（対数目盛）を，縦軸にベースラインに対する F 値の差をとっている．この散布図から，非関連文書の数はおよそ 100 が最適であることが示唆される．

Storm を利用することによる処理速度の向上を示す実験では，並列数が 5 を超えた辺りで実行時間の変化がなくなり，およそ 6 分に収束した．ここで，入力スパウトにより全ての文書を送り出すのに必要とする時間を計測したところ，こちらもおよそ 6 分を要した．このことから，これ以上本システムを並列化する意味はなくなり，同時に Storm を用いて並列数を増加させることにより実行時間を短縮，すなわち処理速度を向上させることができたと言える．

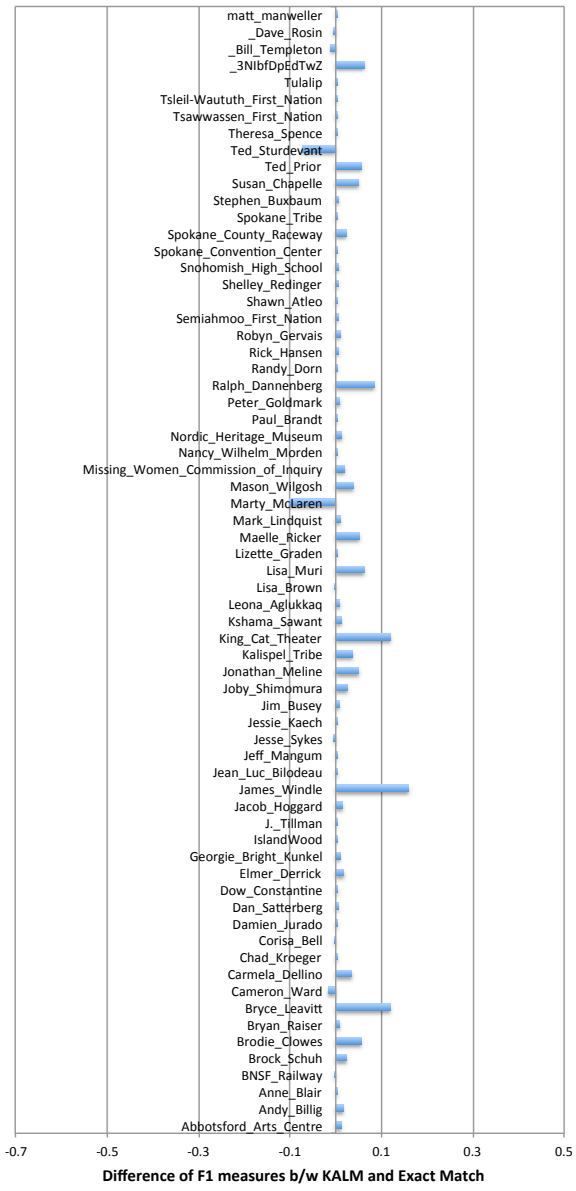

図 4　エンティティ毎の F 値のベースラインに対する差（KALM）

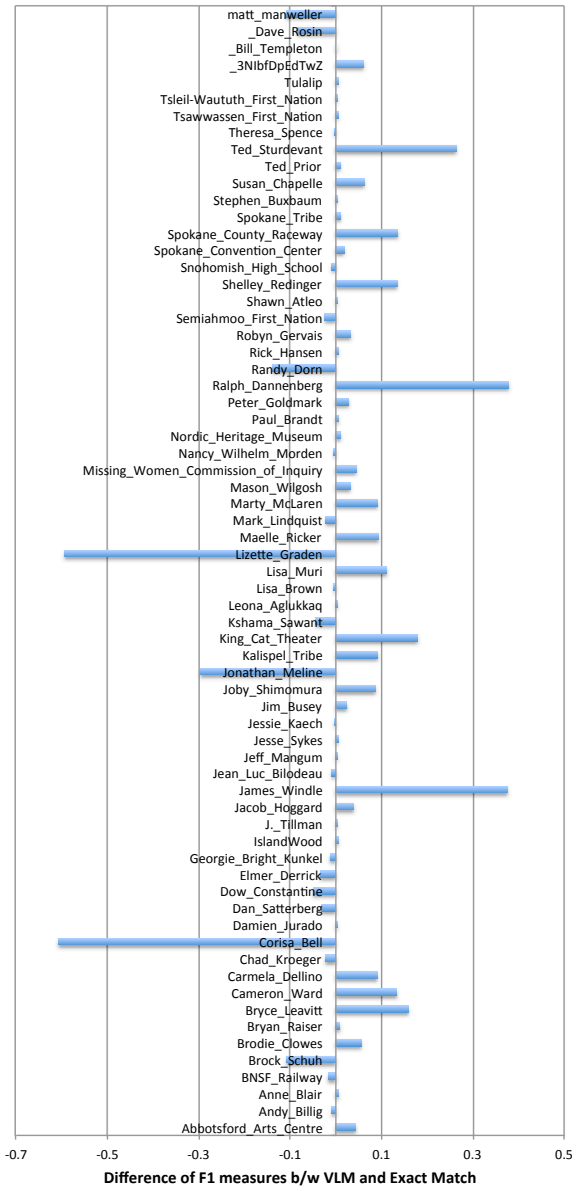

図 5　エンティティ毎の F 値のベースラインに対する差（VLM）

5. ま と め

本論文では，インターネット上の情報を自動的にフィルタリングし，知識ベースの記事の更新が必要となるような新情報を実時間で検出するシステムを提案した．新情報の判断には言語モデリング手法を用いた 2 段階のフィルタを構築した．具体的には，非関連文書を利用して構築した非関連文書モデルを用いて関連文書の判定を行い，知識ベースの記事を言語モデル化した知識ベースモデル，新情報を含む文書における頻出語から構築した重要文書モデルを利用して新情報を含むかどうかを判定した．また，知識ベースモデルと重要文書モデルはシステムにより新情報を含むと判定された文書を利用して更新することで，時々刻々と更新されていく新情報を考慮した．

評価実験では，TREC KBA Stream Corpus を用いてフィルタリングを行い，本システムが新情報を含む文書の検出タスクにおける性能を改善できることを示した．また，並列数を変化させたことによる実行時間の測定実験では，並列数を増やすことで文書が入力される速度と同等の速度での処理を実現し，大規模なストリームデータに対しても適用が可能なシステムであ

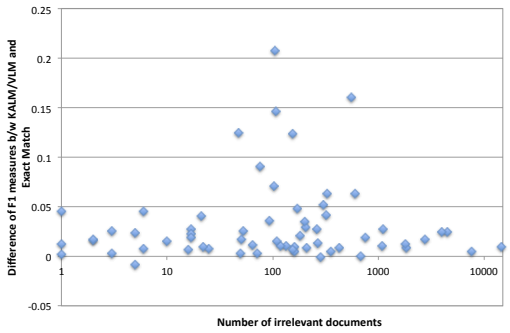

図 6　非関連文書の数と F 値のベースラインに対する差の関係

ることを示した．

今後の研究課題として，システムの性能をより向上させるため，新情報の検出に有効な素性の導入を検討している．本システムで用いた手法は新情報の検出性能の改善に有効であったものの，システムそのものの性能は高くない．このため，新情報の判定により有効な素性を検討し，導入する必要がある．例えば，過去数時間以内にトピックについて言及された文書数，すなわちそのトピックの盛り上がりを捉えることが新情報を含むかどうかの判断に有効であることが考えられる [2] ので，本システムの素性として取り入れることで，更なる性能改善が期待できる．

文　献

[1] R. Abbes, K. Pinel-Sauvagnat, N. Hernandez, and M. Boughanem, “IRIT at TREC Knowledge Base Acceleration 2013: Cumulative Citation Recommendation Task,” in *Proceedings of the TExt Retrieval Conference (TREC)*, 2013.

[2] K. Balog, H. Ramampiaro, N. Takhirov, and K. Nørvåg, “Multi-step classification approaches to cumulative citation recommendation,” in *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, 2013, pp. 121–128.

[3] A. Bellogín, G. G. Gebremeskel, J. He, J. Lin, A. Said, T. Samar, A. P. de Vries, and J. B. Vuurens, “CWI and TU Delft at TREC 2013: Contextual suggestion, federated web search, KBA, and web tracks,” in *Proceedings of the TExt Retrieval Conference (TREC)*, 2013.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the royal statistical society, series B*, vol. 39, no. 1, pp. 1–38, 1977.

[5] L. Dietz and J. Dalton, “UMass at TREC 2013 Knowledge Base Acceleration Track: Bi-directional Entity Linking and Time-aware Evaluation,” in *Proceedings of the TExt Retrieval Conference (TREC)*, 2013.

[6] J. R. Frank, S. J. Bauer, M. Kleiman-Weiner, D. A. Roberts, N. Tripuraneni, C. Zhang, C. Re, E. Voorhees, and I. Soboroff, “Evaluating Stream Filtering for Entity Profile Updates for TREC 2013 (KBA Track Overview),” in *Proceedings of the Text REtrieval Conference (TREC)*, 2013.

[7] J. R. Frank, M. Kleiman-Weiner, D. A. Roberts, F. Niu, C. Zhang, C. Ré, and I. Soboroff, “Building an entity-centric stream filtering test collection for TREC 2012,” in *Proceedings of the Text REtrieval Conference (TREC)*, 2012.

[8] T. Kenter, “Filtering Documents over Time for Evolving Topics–The University of Amsterdam at TREC 2013 KBA CCR,” in *Proceedings of the TExt Retrieval Conference (TREC)*, 2013.

[9] J. Lafferty and C. Zhai, “Document language models, query models, and risk minimization for information retrieval,” in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 111–119.

[10] X. Liu, J. Darko, and H. Fang, “A Related Entity based Approach for Knowledge Base Acceleration,” in *Proceedings of the TExt Retrieval Conference (TREC)*, 2013.

[11] M. F. Porter, “An algorithm for suffix stripping,” *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[12] J. Wang, D. Song, C.-Y. Lin, and L. Liao, “BIT and MSRA at TREC KBA CCR Track 2013,” in *Proceedings of the TExt Retrieval Conference (TREC)*, 2013.

[13] X. Wang, H. Fang, and C. Zhai, “A study of methods for negative relevance feedback,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 219–226.

[14] C. Zhai and J. Lafferty, “A study of smoothing methods for language models applied to Ad Hoc information retrieval,” in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 334–342.