HotMiner:Discovering Hot Topics from Dirty Text

Malu Castellanos

村上浩司

Overview (1)

会社：顧客のホットな情報を特定することは非常に大事。
価値ある情報
・会社にコスト削減の可能性と更なる競争力
・顧客のニーズ

テクニカルサポートセンター
→　サポートエンジニアを削減できる！
顧客のホットな問題を把握→それを解決できる顧客の準備
→顧客はその情報を見つけて自分で問題を解決

サポートセンターのログからのマイニングでホットな情報を発見するアプローチ

Overview (2)

＜*検索ログ*＞マイニングへの手法　：
普通では難しい　→　顧客のホットなトピックを発見できる

＜*ケースログ*＞への手法　：
ホットトピックマイニングのために、より適したケースから
関連した文の抽出を含む。

＜手法＞
一般的な文書（スペルミス、略語、特別な記号、文法間違い、
不可解な表、あいまいな句読点がある）を扱う。

→　・いろんな技術が必要、とくにポストフィルタリング
・語彙を正規化するために「汚い」語のシソーラスを作成
・コードと表を取り除き、重要文同定

Introduction

顧客のニーズの把握は大事だが、その特定は難しい

・文書を階層的にトピックに分類しておく：

→　手動での取り組みはもっとも成功、ヤフーはその例。

とはいえ。。。

労力がかかるし、ドメインのエキスパートがいなきゃだめ。

でも機械じゃかなわない！！！

文書の中身をマイニングせずに、ユーザの情報要求を含む

時系列的な情報であるログを対象

→　・最も興味のあるトピックだけに焦点を絞れる

Introduction

手法：：カスタマーサポートセンターのログからホットトピックを
マイニング (Hewlett-Packard**のログ**)

顧客が問題を抱えたとき

1. **電話する**
2. **ウェブサイトから必要な情報をとってくる**

→ 2**種類のログ** (search、case) **を対象**

Search：**検索文字列の追跡、クリックのパス。**

Case：case**が開いている間の動作、イベント、対話のヒストリ**

ポイント：汚い、さまざまな種類のノイズを扱うこと

→ **有意義な結果を高い信頼性で得るため。**

Introduction

＜Search Log：**自己解決型ユーザのホットトピックを発見**＞

クエリを使って検索し、文書を見ているという仮定
- **ユーザの観点とマッチするホットトピックを発見可能。**
- **ホットトピックはその質が高いときだけ役に立つ。**

…でも：寄り道したり違う文書を見たりすると、ログが複雑に！

そこで…

Postfiltering**法を利用**

→ **各文書の中身を考慮するもの。**
ホットトピック内で、無関係の文書を特定

無関係の文書の特定

→ **高品質のトピックをえることにはさほど有益でない**
ユーザを迷わすノイズになる文書を決め打ち可能

Introduction

＜Case Log：電話などの時の殴り書き、メモのようなログを対象＞

既知の問題のホットトピックの発見

→ このログは汚い！（汚さの種類：本文）

タイポ、ミススペル、略語

まず…不規則さを解決する手法を利用

もしくは…ノイズにロバストな手法

単語の正規化：シソーラスアシスタント（適切な重複した語を特定）に対し文書セットをかませる）

重複：類義語が代表的。でもここではこれらを考慮しない。

なぜなら対象の単語はドメイン依存な専門用語で、

こうした種類の類義語を持たないから

Part I:Mining Search Logs

自分で問題解決する人のホットトピックの発見

1. SearchView,**検索語から検索される文書の閲覧を利用**
   →　顧客の問題のホットトピックと関連文書を発見

2. **文書の中身から導かれる文書の閲覧を利用**
   →　関係ない文書を特定する（結果の適合率を上げる）

＜処理手順＞
プリプロセッシング　→　クラスタリング　→
ポストフィルタリング　→　ラベリング　の順。

Preprocessing (1)

生ログから関係するデータを抽出、マイニングに適した
フォームに文書を変換

データコレクション:
- **ログ::ユーザの行動の完全な情報を提供**
  HP**のカスタマーサポートでは、ログは**3**タイプをもつ。**
  使い方 (Usage)**ログ:ユーザの動作の情報**
  (DoSearch、ReviewResultsDoc、**など**)
  クエリログ: DoSearch**の後に入力されるクエリの情報**
  ウェブログ:開いたウェブページの情報

まず、クエリが投げられた時間情報の取得 (Usage,Query**ログ**) **、**
そしてWeb**ログと関連させて、開いたすべての文書の情報を得る**
→ 文書ークエリ の関係 (search view)

Preprocessing (2)

データクリーニング:
(文書ークエリ) をクリーニングする。エントリのいくつかを除去、
クエリの正規化、特別な記号、ストップワードの除去、ステミング

データ変換:
きれいになったエントリは、ベクトル化される。ベクトルセット (V)
W_{ij}の要素を含む。文書i中の語jの重み。
ホットトピックだけ欲しいので2文書以上で出現する上位nの
クエリ語を素性として選択。句としての複数回出現は考慮しない

クラスタリング:
ベクトルを使ってクラスタリング。ここではk-meansを利用

Postfiltering (1)

SearchView::ユーザは必要に応じた文書を開くことを仮定
→ ユーザは気まぐれ、興味などで、寄り道、不思議なクリック。

クラスタリング結果::必ずしもきれいであるとは限らない
→ この結果だけではまだだめ。
しかも、ユーザは低品質のホットトピックサービスは嫌い。
→ 不要な文書を特定してフィルタリングする必要あり。
クラスタ内のほかの文書と比べて低類似度のものを取り除く

データコレクション:
実際の文書の内容を検索。

データクリーニング:
先ほどのクリーニングと同じように行う。HTMLの場合は余計な情報を削除。

Postfiltering（2）

データ変換（素性選択）：

文書内の単語のProbability Distribution**を計算。少なくとも**2**文書に現れる最も高い確率を持つ**l**個の語を選択。**

$$w_{ij} = k_{ij}[tf_{ij} \log(n / df_j)]$$

*Wij***は、**TF・IDF**を少し変形。***n***は文書数、***tfij***は文書**i**内の単語**j**の頻度、***k***は単語*j*が文書*i*の重要箇所（タイトルなど）に現れるときの**Flexibility to augument。K**はデフォルトで１．**

無関係文書特定：類似度計算とフィルタリング

類似度計算：文書di**とその他**dj**との類似度、コサイン距離で計算**

個々の文書類似度

AVG_SIM(di)：**自分以外の文書との距離の平均を計算**

クラスタ平均類似度

AVG_SIM(Ck)：**各文書の平均距離の平均を計算**

Posftifltering（3）

フィルタリング：
AVG_SIM(*dij*) **は、クラスタ内で、ある文書と他の文書間の関係**
　　高　：　その文書の中身はクラスタにフィット
　　低　：　低ければクラスタと無関係の文書

どれくらい低ければ無関係なのか？？

この閾値はクラスタ平均類似度からの偏差の大きさに依存。
　→　クラスタごとに標準偏差S(*Ck*)**を計算**

AVG_SIM(di)**を**Z**値で正規化**
　Z(AVG_SIM(di)) = [AVG_SIM(di) -AVG_SIM(Ck)] / S(Ck)

Postfiltering (4)

ラベリング:

フィルタリング済のクラスタリング結果がきれい

→ ラベルを人手もしくは自動で各クラスタに付与

人手:クラスタの文書を調べて、ヒューリスティクスで意味的な
ラベルを付与。

自動:半分以上の文書に現れるL個の共通単語がラベル

ラベルがつかないクラスタは破棄される。同じラベルを持つ
複数のクラスがある場合は、マージ、Postfiltering**する。**

Experimental Result (1)

これまで説明したものは、HotMiner**というツールとして作成。**

HP**のログ：登録したユーザに対して**100**万文書が閲覧可能**
- **出来る人なら、自分で解決した方が効率的**
- **出来なければ**case**を開いて解決**

検索プロセス：　(A.**トピックオントロジー　**B.**キーワード検索)**

A：**ユーザの考えと違う**
ドメインの専門家 (サポートエンジニアなど) により適している
(文書集合の中身の深い知識があり、そのトピックに詳しい)

B:**ユーザは情報要求を特徴付ける単語からクエリを作成**
欠点
- **ユーザが欲しい文書ではクエリ内の単語が出現しない**
- **不必要な文書が検索される (→どつぼにはまりやすい)**

Experimental Result (2)

HotMiner：

ユーザのニーズにマッチするホットなトピックをみつけて、それに対応する文章をまとめる。

→ これがWEB上で直接見れるなら、顧客は直接その文書を見つけられる。なので、自己解決は非常に効率的になる。

＜対象データ＞

- **1999年の11月を対象**
- **49，652のファイルをユーザは開き、うち12，678文書は明確**
- **クエリ数は27，884**

80%の顧客の電話は、20%の問題について問題を質問：

→ 問題を特定し、適切な解決文書のウェブサイトを作成する必要

正規化：正規化は適用範囲を限り、手で正規化する。

Experimental Result (3)

＜クラスタリング＞

- SOM_PAKを使用：ビジュアライズがきれい
- マップを作成、(Fig6.3)、セルのラベルがオーバーラップ
  → あるトピックを構成するグループ化をセルが表現
- トピックを発見するためにもう一度SOM
  → 最初のクラスタリングでのグループのセントロイドがわかる
  (このときのクラスタは、サブトピックを含むから)
- しかし、時々グルーピングが意味を成さない
  1. SOMによるセントロイド間の関係が人間の直感と合わない
  2. 同じ単語での異なった使い方があるから。

Experimental Result (4)

＜ポストフィルタリング＞

ベクトルを作るためのTD・IDFを変形した式でのkの値と、内容を考慮する語に依存した方法によってベクトル化

- **(a) タイトル内の非ストップワードの頻度**
- **(b) 文書全体での非ストップワードの頻度**
- **(c) kが1以上としたときの (b) の値、タイトルのみ**
- **(d) 文書に現れるクエリの語**
- **(e) 文書の先頭p語 (問題を解決する文書は先頭に情報がある)**

結果、a,c,eが低い、最も高い精度だったのは (b) 。

Experimental Result (5)

＜ラベリング＞

ラベル：　(A.クエリ語から得る　B.内容語を使って得る)

　Bには2つのオプション

　　・内容語の種類の選択 (非ストップワード、名詞、名詞句)

　　　ＰＯＳタガーや名詞句同定器が必要。Inxightをつかった。

　　・内容語を、タイトルから得る、文書全体から得る

結果

　タイトル中の非ストップワードを使うと、良いラベルを得られた。

　→　タイトルは文書の内容の重要な情報を伝えているため (良要約)

　すべての文書が有益なタイトルではないとき

　→　ラベル付けは文書全体で現れる非ストップワードを使うべき

　　名詞句を使うより名詞を使う方がいい

Experimental Result (6)

「Shared Memory」の問題がトピックのとき
- 「memory」は名詞、名詞句を使うときに割り当て
- 「shared」「memory」は非ストップワードを使うときに割り当て

もし文書がShared Memoryに関連していても…
- 関連ドキュメントのタイトルの殆どは「Memory」が含まれる
- 他のドキュメントに「shared」が含まれる
- 数少ない文書に「Shared Memory」が含まれる。

名詞と名詞句を使うとき
- 「Memory」はラベリング時に使用される。
- 「shared」は名詞でも名詞句でないので対象外。

非ストップワードを使うとき
- どちらもよく現れれば、ラベルとして利用
- ときおり完全に意味があるわけでないラベルを返す。

Experimental Result (7)

Fig6.4 「y2k 10.20patch」
→ “HP-UX10.20 for Y2K compliance”
Fig6.5 **ラベル**“sendmail mail unknown”**は解釈が容易でない、**
sendmail**の問題のみにトピックが関連付くのか？**
トピックとラベルから、そのSOMマップを表示。(Fig6.4 – Fig.6.6)。

＜考察＞
いくつかの得られたトピックは存在するトピックの階層に現れない
- Y2K**のように新たに出てきたからか**
- **サポートエンジニアが他の観点が必要なのか**
- **エンジニアがユーザが気づいている問題を知るすべがないのか**

どのトピックがその時点でホットなのか、どれだけ対応する解決法が存在するのか、限られた情報の中で自己問題解決を試すときのユーザのクリック動作は非常に興味がある。

Experimental Result (8)

＜評価＞

・全部を評価するのは大変なので一部だけ
・１０％のホットトピックをランダムにサンプリング
・そのうちの８０％は有意義。
・２０％は、無意味
　１．我々にドメインの知識がない
　２．SearchViewはいいけどContentViewが駄目なのかが原因

意味のないトピックの削除
　→　意味のないホットトピックは自動的に、2度の削除のチャンス
　・いらないトピックを除くためのAVG_SIM(Ck)、S(Ck)に対する閾値をセットするポストフィルタリングステップ
　・有意義なトピックが見つからないときのラベリングステップ

結局は人が最終的に判断しないと偉そうなことはいえない。

PartII Mining Case Log

Search Log**に対して、**Case log (**電話、**e-mail)

モチベーション：Search Log**内の検索、とクリック情報を**
マイニングするだけじゃだめ

Case

- **非構造化省略、ミススペル、コード、変な句読点などの塊**
- **問題に関する技術的なことだけでなく、対応中の業務についての情報も含む**

→ Case**はホットトピックを見つけるのには汚すぎ**

基本的な考え方：

- **出来る限り変則的なものをきれいにする**
- **ホットトピックをマイニングするためにより適切な短い抜粋をえるため、最も関連性の高い文を抽出する**

重要なのはクリーニングと抜粋の生成

Technical Description

抜粋の生成する手法は大きく3つの部分に分かれる：

1．アプリに非依存なシソーラスアシスタントによるクリーニング
2．アプリ主導の文同定のコード／表削除によるクリーニング
3．抜粋を生成する文抽出器による内容抽出

第1ステージ：：汚い文からの単語の頻度を計る
→ シソーラスは不可欠だが、ドメイン特有の単語は特殊なので
特殊な語の変化の対応を同定する独自のシソーラスを作成
→ アプリケーションに非依存で、手法の最初のステージ

第2ステージ：：クリーニング。でもアプリ／ドメイン主導。ここ
では、要約のツールとcaseのドメインを使ってコードや表を取り除く。

第3ステージ：：文を幾つもの尺度で分析／同定して内容抽出。
最後ステージ：：関連の強い文を抜粋として得る

Thesaurus Assistant (1)

タイポ、ミススペル、省略を除く文書の正規化→シソーラスが必要
構築：：参照語とノイズ語のリストを作成、対応する語を同定
エディットディスタンスを利用。
対応が取れたら、参照語の同義語としてシソーラスに追加
これは人手で行う、自動化はムリ。

＜参照語リストとノイズ語リストの作成＞

1. 文書を個々の単語へ落とす
2. スペルチェック
3. ドメイン特定の辞書にエントリがあるかを確認
4. あれば参照語リストに登録。なければ、ノイズ語として扱う

Thesaurus Assistant（2）

＜適切な対応の獲得＞

スペルミスや省略はエディットディスタンスでカバー
（Smith-Waterman**アルゴリズム**）

→　挿入、削除、入れ替え、並べ替えの編集動作の最小数で距離を決定

距離は0～1の範囲

- **0に近いと対象の2語は離れている**
- **1に近いと2語は似ている。**

閾値を設けるが、これは経験的に決定。

この時点では一つの参照語に、多くの候補語がある。この半分は有効な類義語で、結局閾値は0．67。しかし、4文字以下の単語では上手く行かない、語の長さを考慮すべき？

Thesaurus Creation

一致する語のリストは人手で選別され、シソーラスとなる。

Sentence Identifier

基本的な考え方：　アプリ主導のクリーニング

・抜粋文を構成する関連した文を選別するために、文を同定。
　（この処理は普通の句読点を持つ文を対象）
　　ノイズ文はこの処理は難しい　→　句読点がでたらめだから

・終端記号やプログラミングコードなどをヒューリスティクスを使って取り除いてから分同定を行う。

＜プログラムコード削除＞

プログラムにしか現れないようなコードがある行に注目

・特徴として、短い、特別な文字（＊＝とか）、キーワード（elsifとか）

・このコードは連続した行に書かれているはず

こうした行を取り除く

Sentence Identifier (2)

＜表削除＞

コードの他にも、表情報も削除すべき情報特有の特徴に着目

- **語の間の複数のスペース**
- **カラムをそろえるためのタブコード**
- **それらの連続**

→ 文書をパーズ、語の位置や長さなどを得て、それらの特徴のある連続した行を削除する。

＜文境界同定＞

コードと表が取り除かれた後は、文境界の同定

非言語的アプローチをつかう。

- **句点、？、！があり、次の文の先頭が大文字なら、文とする**
- **行数の制限を設けて、長すぎる文は句点などで区切る**

Sentence Extractor (1)

文の関連性を分析
関連性を示唆する文のアスペクトは多彩
- **文中の語**
- **他の文との相関**
- **文の位置　など**

→　重要な語かキーワード、重要語の相関の同定が重要

1．キーワードの密度での文スコアリングに使う、ワード集合を生成
語の相関による文スコアリング手法は、複数文書間の類似度を計算するIRの手法を1文書に落としたもの。

2．文のグローバルスコアを計算
重み付関数にそれぞれの手法から得られるローカルスコアをたし合わせる。(P149　式)

Sentence Extractor (2)

＜キーワード生成器＞

キーワード集合のフォーマット：(キーワード、{尺度})

- 尺度は、キーワードを同定する手法のセット
- すべてのキーワードが等しく重要ではないので、文スコア計算に影響するように、違う重みを割り当てる。尺度ごとのスコアを足しこむ。(P149　式)

出力の特徴に合わせて重みを調整できる。もし大きなサンプルセットがあれば統計的な手法が適用可能では？

＜テーマとなるキーワード＞

- テキストに頻繁に現れるが
- データの至る所に共通でない単語　→　特徴のある単語

文内のキーワード密度は、文の重要性を示す

ここではTF/IDFを使った。ここでも閾値を設定するが、その最適値は実験しだい。

Sentence Extractor (3)

＜重要位置キーワード＞

重要な文は、重要な位置で使われる単語を含むという仮定
(タイトルと文書の先頭は重要な位置、など)
→ caseでは、先頭部分は単にログ中の行動の種類を示すだけ
→ 重要な先頭部分をもつ章の実際の内容を重要な位置
このような章は関連する語を明確に含む (例えば、こうした章中に位置する非ストップワードがキーワードである、など)
しかし、あまりに非ストップワードが多い場合は棄却する必要

＜手掛かり句＞

確からしい文の関連は、「効率的な」「不可能な」といった単語の出現に影響を受ける、という仮定。

- これらの語や句をボーナス単語、否定的なダメ単語として分類。
- ドメインに依存しているので自動がダメなら人手での調査を必要
- キーワードのフォーマット中の {尺度} に対して、プラス、もしくはマイナスに働く。

Sentence Extractor (4)

＜単語相関：意味的類似度＞
文間の類似度関係を発見するためのエンティティレベルの手法

・文書の文間の文書内部リンクを生成するハイパーテキストリンク生成アルゴリズムで文書内リンクのために使われるIR技術を適用

文書内部リンク
文間での語彙の重なりによる意味的な類似関係を表現
(重みつきの2つの単語間の類似係数の計算により得られる)
もし類似度が十分に大きければ文の間で語彙はマッチ
→ 2つの文は意味的に関連、意味リンクが張られる。
・文のリンクの数はその文がどれだけ中心的かを表す
・よくリンクされている文は文書の内容をよく表す。
・文に割り当てられるスコアはリンクの数に比例。(P151式)

Experimental Results

5000 case文書を自動的に抜粋を生成するために適用

結果の分析は全部人手ではできない
→一部だけ、15 caseをランダムで選んだ。

評価は2つのレベル：
・技術的内容を抽出するツールとしてのパフォーマンス評価
・関連した内容を抽出するパフォーマンスを評価

Mining Case Excerpts for Hot Topics

5つの代表的な抜粋が取得 (Fig.6.9)
　→　既知の問題でのホットトピックをマイニングするために利用
仮定：これらの文はcase文書中での問題の本質を伝えていること

そのままのcaseはムリ、タイトル情報、特定の章の情報でもcaseの特性から難しい。なので、代表となる文からホットトピック発見を行う

SearchLogに対する処理と同じものを抜粋を発見するのに使う。
次の段階はクラスタリング (SOM) の適用、そしてラベル付け。

評価においては、Searchlogのときと同じ問題である、ドメイン知識の必要性という問題にあたった。抜粋が意味的にトピックに沿うかを決めるために、caseの内容を理解しないといけないが、無理。
そこで一部だけを評価、F-measureは、再現率が計算不可能なので無理。平均の適合率は75%。

Conclusions

企業は、顧客の関心、ホットトピックを掴むことが非常に大事
→ ・Hotminerを作成、自動的にカスタマーサポートセンターの Search、Caseログからホットトピックをマイニング
・HPのログに対して適用、汚い（不規則な）データを扱う

Search Log：**すでに開いている文書などを手掛かりにマイニング**
Case Log:**データクリーニングが重要、その後文抽出**

評価
・ノイズの多いテキストからまあまあのレベルのホットトピックを発見する能力がある
・より深い評価はまだ行っているところ
・特定のドメインに対して設定やパラメータの最適化もこれから
・顧客の問題で流行るものを予め特定できたらいいなぁ