twitterの様々な推薦機能で活用されてるらしい、ユーザ-ユーザのフォローネットワークからコミュニティ埋め込み表現を生成するSimClustersの論文を読んだ

n週連続 推薦システム関連の論文読んだシリーズ13週目(毎週水曜更新)

AUTHOR
NewsPicks Data/Algorithm unit インターン 森田大登

PUBLISHED
April 24, 2023

この論文を選んだ経緯

- 2023/04/01頃に、twitterの推薦アルゴリズムの一部(“For Youフィード” -ツイート推薦の機能)が、ソースコード + techブログによる解説という形で公開されました。
- その中で今回は、twitterの様々な推薦機能で共通基盤として活用されているらしい **SimClusters** という手法によるユーザ + 各種コンテンツの埋め込みベクトルの生成手法に注目しました。
- SimClutersは、**ユーザ-ユーザのフォローネットワークからコミュニティを発見**して、コミュニティ埋め込みベクトル(SimCluters表現)を作ります。(SNS特有の手法...? 🤔)
- NewsPicksは経済ニュースサービスの性質に加えてSNSの側面もあるので、ユーザやキーワードフォローの情報の活用可能性もあるのでは...? 🤔
- また、現在NewsPicksでは関連記事やプッシュ通知パーソナライズ等、複数の推薦機能が稼働中or予定なので、“共通基盤”という言葉に惹かれました。

参考:

- SimClustersの論文
- 公開されたtechブログ
- 公開されたgithub リポジトリ

SimClusters論文の基本情報

- title: SimClusters: Community-Based Representations for Heterogeneous Recommendations at Twitter.
- (“for Heterogeneous Recommendations” 性質の異なる複数の推薦機能達の為の…😉)
- published date: August 2020,
- authors: Wondo Rhee, Sung Min Cho, Bongwon Suh
- url(paper): https://www.kdd.org/kdd2020/accepted-papers/view/simclusters-community-based-representations-for-heterogeneous-recommendatio

どんなもの?

- twitter の **SimClusters (Similarity-based Clusters)**というコミュニティ埋め込みベクトルを生成する手法の論文.
- 「K-POP」や「機械学習」といった共通の話題や、「職場が一緒」「高校が一緒」といった社会的関係から構成されるTwitter上のコミュニティ（約 10^5件）を発見し、ユーザやコンテンツと各コミュニティの関連を表す埋め込みベクトルを生成する.
- 様々な推薦タスクに活用できる、汎用的な埋め込み表現であるとの事.(論文の後半のオンライン実験にて、活用法を紹介していた.)

SimClustersのモチベーション

- Twitterでは多種のアイテムに対して推薦機能がリリースされている:Tweet, Event, Topic(NPにおけるキーワード的な位置づけらしい), ハッシュタグ, ユーザ
- 推薦アイテム達はそれぞれ異質(Heterogeneous)である：Item Cardinality(=計算のスケーラビリティに影響を与える)とShelf Life(推薦生成のlatencyに制約を与える. 賞味期限的なもの)が異なる(table 1参照)
- これまでTwitterは、**これらの異なる推薦問題に個別に対応するシステムを構築しており、再利用や共通化はほとんど行われていなかった。**

Recommendation Problem (i.e., target)	Item Cardinality	Shelf Life	Display Location
User recommendations	$\sim 10^9$	Weeks	Who To Follow
Topic Tweet recommendations	$\sim 10^8$	Hours	Home, Explore
Out of network Tweet recommendations	$\sim 10^8$	Hours	Home
Similar Tweet recommendations	$\sim 10^8$	Hours	Tweet details page
Event recommendations	$\sim 10^3$	Hours	Explore
Trending Hashtags	$\sim 10^5$	Hours	Explore

Table 1: A partial list of recommendations problems at Twitter along with the number of possible recommendable items, the shelf life of the recommendations, and where they are shown on Twitter.

Shelf Life が異なるケースって例えばなんだっけ

- フォロワーグラフ(フォロー関係のネットワーク)は比較的ゆっくりと変化するため、ユーザフォローの推薦は数週間にわたってユーザ嗜好との関連性を保つことができる(したがって、**バッチ処理で計算することができる**)
- 一方で、ツイート推薦は陳腐化するのが早い(i.e. lifespanが短い🤔)為**オンラインシステムでリアルタイムに近い形で生成する必要があり**、この場合、バッチ計算では要件を満たせない.
- NPの例で言えば、通常の経済ニュースはShelf Lifeは短め。(ツイートよりは長そうだが、shelf lifeは2~3日?🤔)
- NPオリジナル記事やオリジナル動画、トピックス(=専門家による記事)等は、もっとShelf life長い気がする🤔
- ユーザやキーワードは更にshelf lifeが長いはず🤔

SimClustersのモチベーション

目的:

- 異質(Heterogeneous)な全て or ほとんどの推薦機能の性能を高めるのに役立つ共通基盤的なシステムを構築したい...!

解法

- **ユーザ-ユーザグラフからコミュニティ埋め込みベクトル**を作る。

- 各コミュニティは、そのコミュニティ内の多くのユーザがフォローしているインフルエンサー集合によって特徴づけられる。
- 異質(Heterogeneous)な推薦コンテンツ(i.e. table 1のターゲット)のそれぞれは、**コミュニティ空間におけるベクトル**として表され、アイテム j に対する i 番目のコミュニティに対応する要素は、i番目のコミュニティがアイテム j にどれだけ興味を持っているかを示している。
- 最終的には、**異質な推薦対象を、同じ空間のスパースで解釈可能なベクトルとして表現でき**、様々な推薦やパーソナライゼーションタスクに活用可能。

技術や手法の肝は?

SimClustersシステム(図1参照)は、2つのstageで構成されている：

- stage 1: Communityの発見 (User Interest Representationsの取得)
- stage 2: 各種 Item Representationsの取得

技術や手法の肝は?

SimClustersシステム(図1参照)は、2つのstageで構成されている：

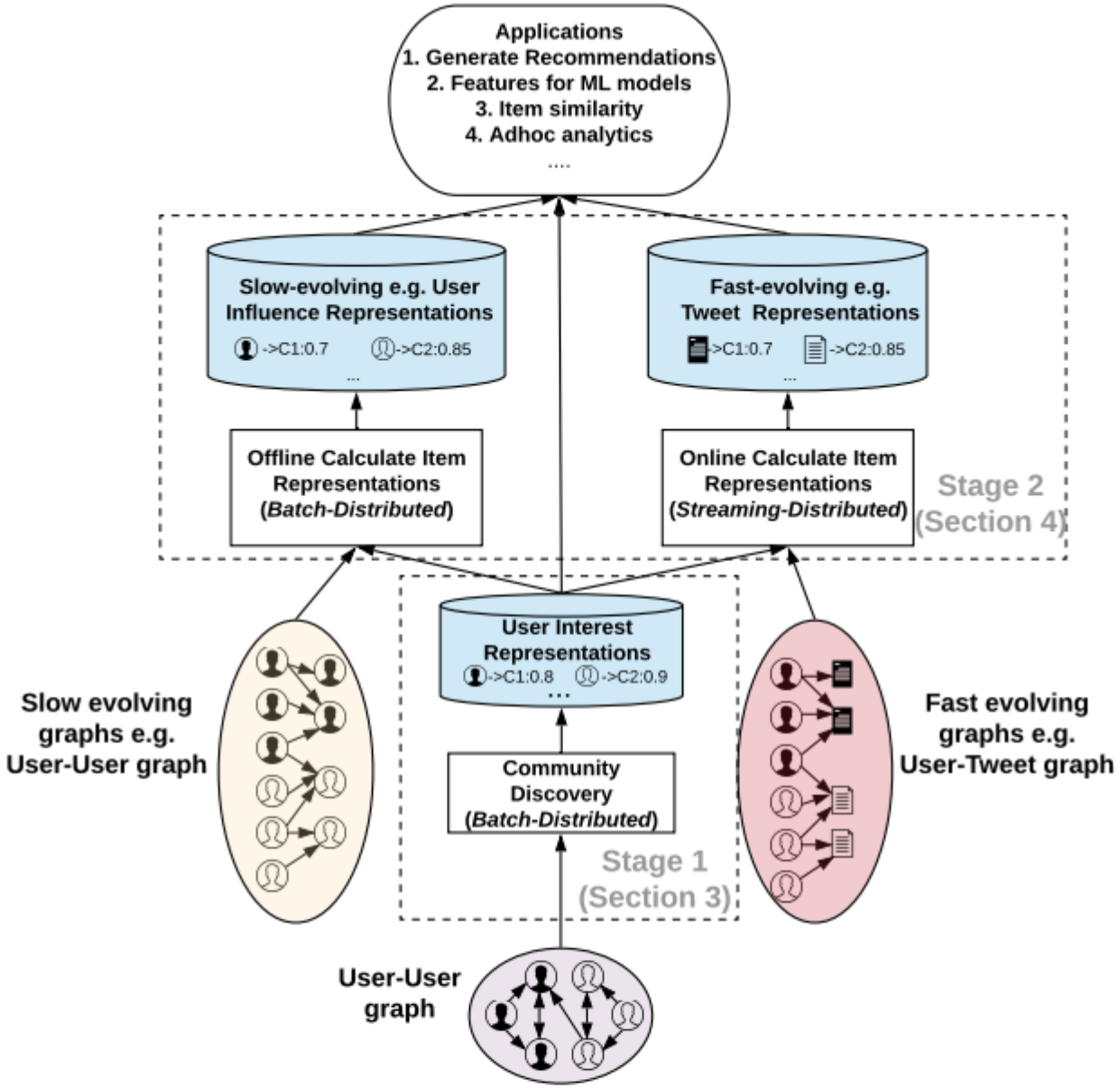


Figure 1: Overview of SimClusters. Stage 1 detects bipartite communities from the user-user graph, which is fed to Stage 2 which runs several item representation jobs in parallel. For

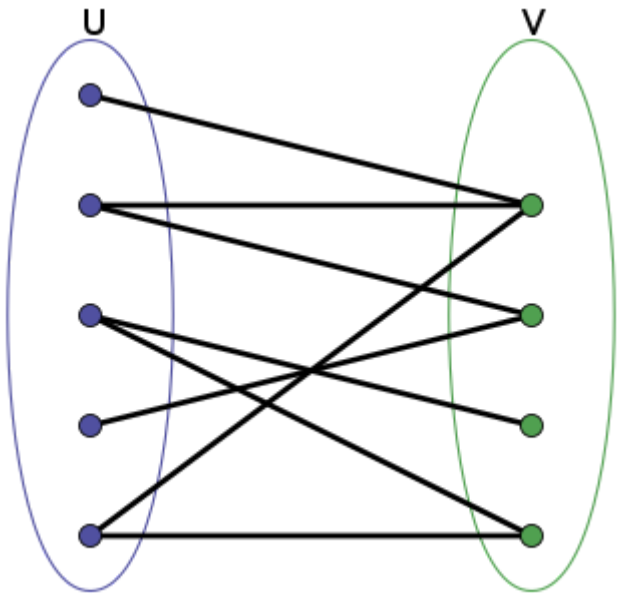
illustration, the color of the users and items in this figure indicates what community they get assigned to. In actuality, both users and items can participate in multiple communities.

stage 1: Communityの発見 (User Interest Representationの取得)

- stage 1 は、Twitterのユーザとユーザのグラフからコミュニティを発見する
- ここで言う"グラフ"とは、**ユーザ間のフォロー関係を表す有向グラフ**(directed graph)の事。
- 代表的な既存研究に習って、有向グラフを**二部グラフ(bi-partite graph)として再定義**して扱う

二部グラフ(bi-partite graph)ってなんだっけ?

> 数学、とくにグラフ理論における2部グラフ(にぶグラフ、英: bipartite graph)とは、**頂点集合を2つに分割**して、**各部分の頂点が互いに隣接しない**ようにできるグラフのこと. 互いに隣接しない頂点からなる集合を**独立集合**といい、頂点集合を n 個の独立集合に分割可能なグラフのことを n 部グラフ (n-partite graph) という. (wikipediaより)



stage 1 における問題設定を定義:

- 左の独立集合Lと右の独立集合Rを持つ**user–user二部グラフ**(=各頂点がユーザを意味する二部グラフ)が与えられるとする.
- そのグラフからk**個のコミュニティを発見** し、各左ノード(=Lの各頂点)と右ノード(=Rの各頂点)にそのメンバーシップの強さ(=**k個の各コミュニティに所属する度合いの強さ**)を示す重みをつけてコミュニティに割り当てる。

ユーザフォローの有向グラフを二部グラフとして再定義する利点

- ユーザ->ユーザのフォロー関係を表す有向グラフを、二部グラフとして再定義する一つの利点は、R を L と異なるように選択できること。(i.e. 全ユーザ集合を 2 つに集合に分けられる事?🤔)
- 特に、典型的なソーシャルネットワークの**大部分の辺は少数のユーザに向けられている**(=多くの一般ユーザがインフルエンサー的なユーザをフォローしている状況🤔)ので**、L よりも小さな R を選ぶことは理にかなっている(グラフからコミュニティを発見する上で🤔).
- Twitterの場合、最もフォローされている上位10^7人のユーザ(=インフルエンサー)を R に含め、残りの全てのユーザを L に含める.
- (文脈から判断すると"有向グラフ→二部グラフに再定義"というのは、まずフォロー関係の有向グラフからRとLの2つのユーザ集合を定義する。次にR内 & L内のノード間のエッジ=辺を切り、二部グラフとして再定義させる、という方法っぽい🤔)

定義したstage1の問題をどんなアプローチで解く?

user–user二部グラフからk個のコミュニティを発見し、各ユーザに対してk個のコミュニティに所属する度合いの強さを表す埋め込み表現を取得する為に、次の**3 step** のアプローチを採用している。

- Step 1. Similarity Graph of Right Nodes: 右の独立集合Rについて、**類似度グラフ(Similarity Graph)**を取得する.
- Step 2. Communities of Right Nodes: Rの類似度グラフから、**k個のコミュニティを発見**する(=Rのユーザ達をk個のコミュニティに割り当てる).

- Step 3. Communities of Left Nodes: 左の独立集合Lの各ノード(=頂点=ユーザ)を、Step 2で発見した各コミュニティに割り当てる(=**ユーザと各コミュニティとの関連度合い**を埋め込む).

定義したstage1の問題をどんなアプローチで解く?

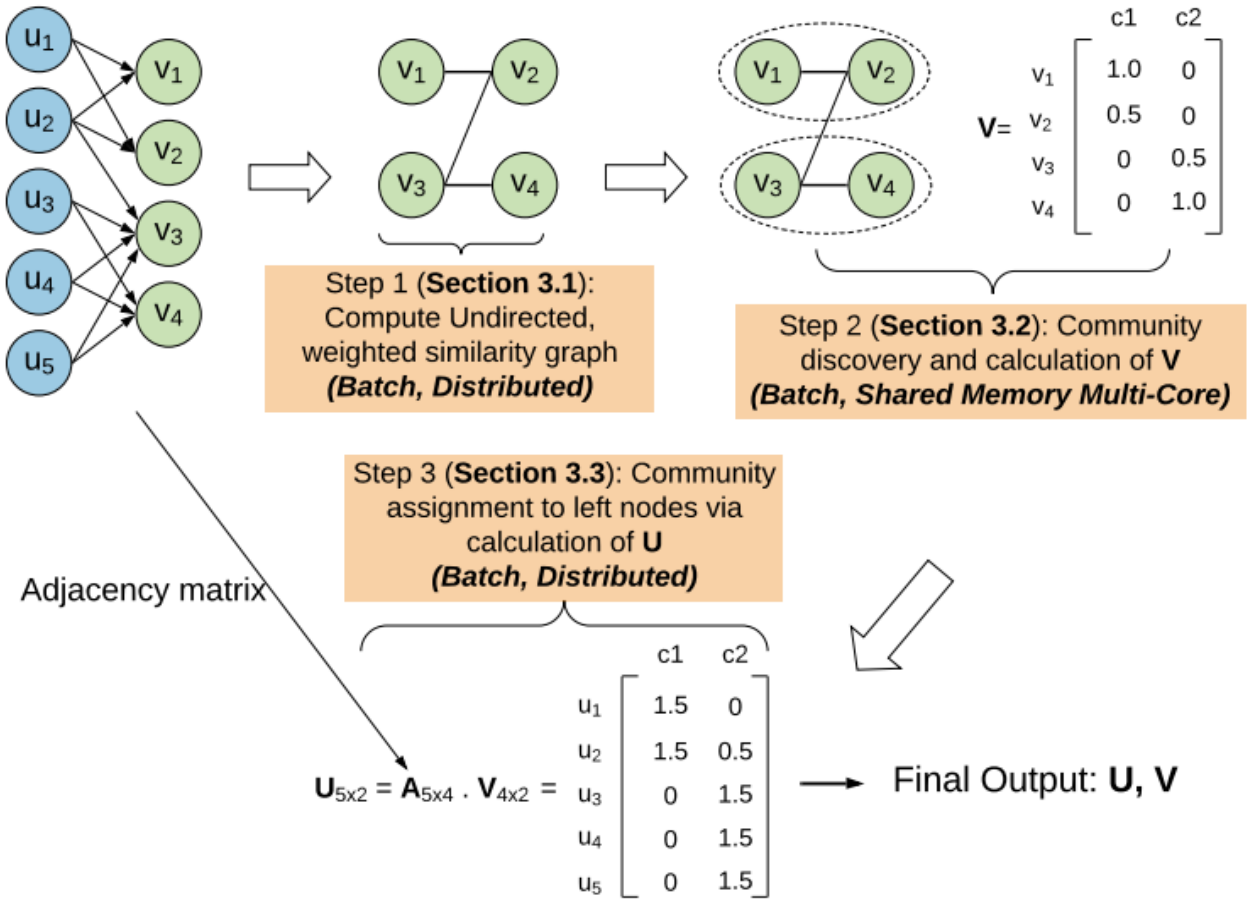


Figure 2: Overview of 3-step approach to discovering bipartite communities (Stage 1).

stage1-step1: 右の独立集合RのSimilarityグラフを作る ①作り方

- このステップの目的は、右の独立集合Rのノードを用いて、より小さな uni-partite(一部)の無向グラフ G を構築する事.
- R 内の2つのユーザ(u, v)間の重み(=similarityとして表される)を、二部グラフの左側(独立集合 L)の"フォロワーのコサイン類似度"に基づいて定義する.

$$similarity(u,v) = \vec{x_u} \cdot \vec{x_v} / \sqrt{|\vec{x_u}||\vec{x_v}|}$$

- "フォロワーのコサイン類似度"について具体的には、ユーザuとvについての"フォロワーのbinary incidence vectors(入射ベクトル)"をそれぞれ $\vec{x_u}$ と $\vec{x_v}$ とする(=長さは左集合 L の大きさ.もしLのユーザiがuをフォローしていれば i番目の要素が1, それ以外が0であるようなbinaryベクトル).

stage1-step1: 右の独立集合RのSimilarityグラフを作る ②dense過ぎるsimilarityグラフを生成しない工夫

- 定義によれば、2つのユーザは、二部グラフにおいて一人以上の共通の隣人(L内のフォロワー)を共有するだけで、非ゼロの類似度がある.(=similarityグラフを行列で表した時の非ゼロ要素!)
  - i.e. similarityグラフ G にエッジ(=ノード間の接続=辺)を持つことになる.
- 極端に密な similarityグラフ を生成しないために、similarityスコアがある閾値より低いエッジ(=接続)を破棄する。
- さらに、各ノードのsimilarityスコアが最も大きいノードを最大で一定数のみ保持するようにする。

stage1-step1: 右の独立集合RのSimilarityグラフを作る ③このステップの利点

- twitterでの運用においてstep1は、10^9 個のノードと〜 10^{11}個のエッジを持つ有向/二部グラフを入力とし、10^7 個のノード(=独立集合Rの要素数)と 10^9 個のエッジを持つ無向のsimilarityグラフを出力する。

- つまり、**shared-nothing のcluster-computingスケール**から、**shared-memory のmulti-coreスケール**に変換する事ができた。
- (=>step1によりグラフのスケールが大幅に削減された事で、複数のリソースにまたがる分散処理ではなく、一つのリソース内で共有メモリを使用するようなマルチコア処理で計算可能になった、って事か🤔)

stage1-step2: RのSimilarityグラフから、k個のコミュニティを発見する

- step1で得られた無向のSimilarityグラフを用いて、step2では、**密に接続されたノード集合のコミュニティを発見する**ことを目的としている.
- コミュニティ発見アルゴリズムの長い歴史にもかかわらず、twitterのスケーラビリティ要件を満たすことができる既存のソリューションを見つけることができなかった.
- 今回の要求を満たすために、**Neighborhood-aware Metropolis Hastings(以下、Neighborhood-aware MH)**というアルゴリズムを開発し、similarityグラフからk個のコミュニティを発見し各Rノードを割り当てる。
- (ココを話すと時間が足りないので今回は割愛します...!気になった方は[このqiita記事]に、pythonによるテストコードと実装を載せています)
- (ベイジアンアプローチの文脈で聞いたことある"Metropolis-Hastings"なので、サンプリング的な事を行って目的関数を元に最適なk個のコミュニティを探索する...!みたいなアプローチ🤔)

stage1-step3: 左の独立集合Lの各ノード(=頂点=ユーザ)を、Step 2で発見した各コミュニティに割り当てる①

- step2の出力は、右集合R(i.e. インフルエンサー達)に対するコミュニティ割り当て行列 $V_{|R|\times k}$ である. (i 行目の行ベクトルは、右ノードiに割り当てられたコミュニティを示すbinary要素のベクトル.)
- stage1の問題設定において残りの問題は、左集合Lに対するコミュニティ割り当て行列 $U_{|L|\times k}$ を取得する事.

stage1-step3: 左の独立集合Lの各ノード(=頂点=ユーザ)を、Step 2で発見した各コミュニティに割り当てる②方法

- シンプルに、Lの各ノードの**neighbors(=全て右集合Rのノードのはず!)が割り当てられているコミュニティを集約**する事によって、各左ノード(=一般ユーザ)をコミュニティに割り当てる:

$$U = truncate(A \cdot V)$$

ここで、$A_{|L|\times|R|}$ を**入力二部グラフの隣接行列(adjacency matrix)**(要素=1だと隣接してる事を示すbinary行列.)とする. $truncate()$ 関数は、**任意の1ユーザが所属するコミュニティの数を制限するfunction**(ストレージを節約するため).

得られた$U_{|L|\times k}$を **User Interest Representations** と呼び、Uを入力としてstage2にて、各種アイテムのSimClusters表現を取得する。

stage 2: 各種 Item Representationsの取得①計算方法

- stage2では、ツイート、ハッシュタグ、ユーザなど、**様々な推薦問題の対象となりうるアイテムに対するSimCluters表現**を計算する。
- stage2の一般的な枠組みは、**アイテムに関わったすべてのuser interest Representationsを集約**してアイテムの表現を計算すること(なるほど…!シンプル!). つまり、アイテム j のSimClusters表現は、以下.

$$W(j) = aggregate(U(u), \forall u \in N(j)) \quad (3)$$

ここで、$N(j)$ は、対応するユーザ-アイテム二部グラフにおいてアイテムjに関与したすべてのユーザを示し、$W(j)$ と $U(u)$ はいずれもベクトルである。本手法では、$aggregate()$**関数として"exponentially time-decayed average(指数関数的時間減衰 平均)"を選択**。これは、アイテムに関わったユーザの貢献度を、そのユーザがアイテムに関わった時間に基づいて指数関数的に減衰させる。(半減期はアイテムのshelf-lifeに応じて設定.)

stage 2: 各種 Item Representationsの取得 ②スケーラビリティの工夫

- aggregateの結果として得られるベクトルWはUよりもはるかに密度が高く(=>Uはsparseが、Wの場合はdenceになりうる🤔)、その非ゼロ値をすべてスケールで保存することは有益ではない。
- Wのview (指標、サブセット、要約統計量的なイメージ?)を2つ追加する。

    - first view はRで、$R(j)$は**アイテムjのトップkコミュニティを追跡する**.
    - second viewはCで、$C(c)$は**コミュニティcのトップkアイテムを追跡する**.
- 賞味期限が長いものの場合、W、R、Cの計算はbatch処理で行える.
- 賞味期限が短いものの場合、W、R、Cの計算は、incremental更新を行う.(時間減衰 平均的なアプローチなので、"現在の平均値"と"最後に更新されたtimestamp"さえ保存していれば、新しいユーザとアイテムのengagementログを受け取ってWを更新できる...!🤔)

stage 2: 各種 Item Representationsの取得 ③最終的な成果物

最終的には、stage1で得た**user interest表現**に加えて以下のSimClusters表現Wが得られる.(各Wについて、サブセットとしてRやCも保持して活用する。)

- **ツイート表現**: ツイートがどのコミュニティと関連性が強いかを表す埋め込みベクトル
- **トピック表現**: (上に同じ)
- **トレンド表現**: (上に同じ)
- **user influence表現**: ユーザがどのコミュニティで影響を発揮しているかを表すベクトル.
    - user interest表現(=stage1の出力)は、ユーザがどのようなコミュニティに興味を持っているかを表すベクトル.
    - インフルエンサーのuser interest表現Vよりも密である為、優れているらしい。

twitterにおける活用例 ①ツイート詳細ページでの類似ツイートの推薦

- SimClustersの前は、作者の類似性(ページ上のメインツイートの作者と多くのフォロワーを共有するユーザーが書いたツイート)にのみ基づいてツイートを検索していた.
- **ツイートのSimClusters表現**に基づく類似ツイートを追加した. i.e. SimClusters表現(=どのcommunityと関連度が高いかを表すベクトル)がページ上のメインツイートの表現と高いコサイン類似度を持つツイートを取得した. -> ツイートに対するエンゲージメント率が25％高くなった。
- 更に、SimClusters表現の２つ目の活用を追加: SimClusters表現が、ページ上のメインツイートの著者の**user influence表現**(=ユーザがどのコミュニティで影響を発揮しているかを表すベクトル)と高いコサイン類似性を持つツイートを

検索する. -> 更に全体のエンゲージメント率が7%向上。

twitterにおける活用例 ②Home フィードにおけるツイート推薦

- SimClusters以前は、閲覧ユーザのフォローユーザ達が「いいね」を付けているツイートを主なcandidate sourceとしていた。
- **ツイートのSimClusters表現**を使って、新たに 2 つのcandidate sourcesを構築し、A/Bテストで新しいcandidateのエンゲージメント率が33%高い結果となった.
  - 1つ目: ツイートのSimClusters表現が、**閲覧ユーザのinterest表現**とのドット積が最も高いツイートを特定する.
  - 2つ目: 閲覧ユーザが最近「いいね」したツイートと類似するツイートを特定する.(類似ツイートと同様のアプローチ.)
- candidateの生成とは別に、ユーザやツイートのSimClusters表現を、candidateをソートするDNNランキングモデルの**特徴量として追加**する事も有効だった。(CTR等を目的変数とした教師有りMLモデル)

twitterにおける活用例 ③トレンドランキングのパーソナライズ

- Trends(ハッシュタグ、イベント、ニュース速報等の表示)の実装は、Trends候補の検出とランキングの2段階を踏んでいる。
- **トレンドのSimClusters表現**を使って、user interest表現とリアルタイムのtrend表現のdot product(内積!)を使用することで、与えられたユーザ-トレンドペアをスコア化した.

twitterにおける活用例 ④Topic Tweetの推薦

- 「ファッション」や「マーベル映画」など、あらかじめ定義されたトピック(NPにおけるキーワードの様な概念🤔)に関するコンテンツを表示する機能。
- SimClusters以前は、人間の専門家がキュレーションしたテキストマッチングルールに依存して、トピックツイートを識別していた(要するにルールベースの手法だった🤔)
- 多くのご検出(主にツイートのテキストがトピックのルールと偶然一致することによる)があったらしい。
- ->今回SimClusters表現を用いて、**対象トピックのSimClusters表現**と高いコサイン類似度を持つツイート表現を特定する. 次に既存のテキストマッチング規則を適用した. (**二段階にしたって事か**🤔)

twitterにおける活用例 ⑤ユーザフォローの推薦

- Who To Followレコメンデーションの候補は、エンゲージメント予測モデルを用いてランク付けされる.
- このモデルには、**閲覧ユーザと候補ユーザのSimClusters表現**に基づく新しい特徴量が加えられている.(特徴量としての活用.)
- A/Bテストでは、これらの新たな特徴量を使用することで、フォロー率が7%向上することが確認された.

twitterにおける今後の活用 ①通知品質フィルター

- Twitterプロダクトの重要なタスクとして、罵倒やスパム的なリプライやメンションを受けないようにユーザを保護することがある.
- **ユーザとユーザのブロックグラフ（あるユーザーが他のユーザーをブロックした場合）に基づく新しいユーザ表現をSimClustersで開発**し、この表現を特徴として、罵倒やスパムのような返信をフィルタリングするモデルを学習した.
- オフラインテストでは、PR-AUC が 4%向上するという素晴らしい結果を示した.

twitterにおける今後の活用 ②リアルタイムイベント通知

- リアルタイムイベント通知とは、大きなニュースが起こったときに、興味を持ちそうなユーザに通知すること.(プッシュ通知みたいなこと?🤔)
- イベントのSimClusters表現(これは、そのイベントに関する人間が作成したツイート表現をaggregateすることで得られる)を使って、**そのイベントに興味を持つユーザのコミュニティを特定**し、そのコミュニティに興味を持つユーザをターゲットにすることができる.
- (新規アイテムの場合は、そのアイテムの属性情報と類似したツイート表現や、作成者のUser表現を使ったら、aggregateして**新規アイテムのSimCluters表現**を作れるだろうか…?🤔)

まとめと感想

- Twitterの多種の推薦機能で活躍してるらしい、SimClutersの論文読んだ。

- 異質な推薦機能で使える、共通基盤的な埋め込みベクトルは便利そう...!🤔
- NPは経済ニュースサービスだけでなくSNSとしての側面もあるので、ユーザフォローの情報から"ユーザの嗜好"を抽出できるのでは?? 🤔
- 特に、NPの場合は**プロピッカー**(公式コメンテータ的な専門家ユーザ)という概念があるので、「このプロピッカーをフォローしてる」情報は活用できるのでは...!🤔
- ただ、NPの場合はユーザフォロー機能を使ってるユーザが少ない印象. コメントへの"いいね"や"pick"のリアクションをユーザフォローの代替としてネットワークグラフを作成できるのでは...?🤔

参考:

- SimClustersの論文
- 公開されたtechブログ
- 公開されたgithub リポジトリ
- n週連続 推薦システム関連の論文読んだシリーズ13週目(毎週水曜更新):twitterの様々な推薦機能で活用されてるらしい、ユーザ-ユーザのフォローネットワークからコミュニティ埋め込み表現を生成するSimClustersの論文を読んだ