進捗報告

氷見夏輝

富山県立大学
u220051@st.pu-toyama.ac.jp

December 8, 2025

平井さんのあげていた課題

今後の課題

実行時間の短縮

- マルチプロセスや分散処理を用いたスクレイピング
- 近年開発された Vibrato のような高速な分かち書き手法

外れ値に強いクラスタリング手法の検討

- 外れ値が大きいと，クラスタ数の決定がうまくいかず大雑把な分類になってしまう．そのため，k-means 以外の重心を用いるクラスタリング手法を検討する必要がある．

3D グラフに表示されている単語の表示方法

- 現在はすべての単語に対して一定の大きさと色で表示しているが，単語の出現頻度などによって，大きさや色を変えることで，より効率的な探索が行えると考える．

図 1: 画像 A

進捗

課題

実行時間が長すぎる.
外れ値が大きいと，クラスタ数の決定がうまくいかず大雑把な分類になってしまう．そのため，k-means 以外の重心を用いるクラスタリング手法を検討する必要がある.
現在はすべての単語に対して一定の大きさと色で表示しているが，単語の出現頻度などによって，大きさや色を変えることで，より効率的な探索が行えると考える.

進捗

平井さんは k-means という手法で実行を行っていた。その部分を k-medoids という手法で実行できるようにした。
2D グラフに対して機能を追加した.
卒論の構成を考えた.

提案手法, クラスタリング手法の違い

K-means

k-means ではクラスターの中心は「centroid」と呼ばれ、クラスター内のすべてのポイントの平均として計算される。
「centroid」は実際のデータポイントである必要はなく、仮想的な点となることが多い。

K-medoids

k-means ではクラスターの中心は「medoid」と呼ばれ、実際のデータポイントから選ばれる。
「medoid」はクラスター内の他のすべてのポイントとの平均距離が最小となるデータポイント。

提案手法, クラスタリング手法の違い

メリットとデメリット

k-means では外れ値があるとそれに引っ張られて中心がずれることがあるが、k-medoids では実際のデータ点をクラスタ中心に選ぶので外れ値に強くなる.
k-means では, 計算効率がよい, k-medoids では計算コストが高く, 大規模なデータでは効率が悪くなる.

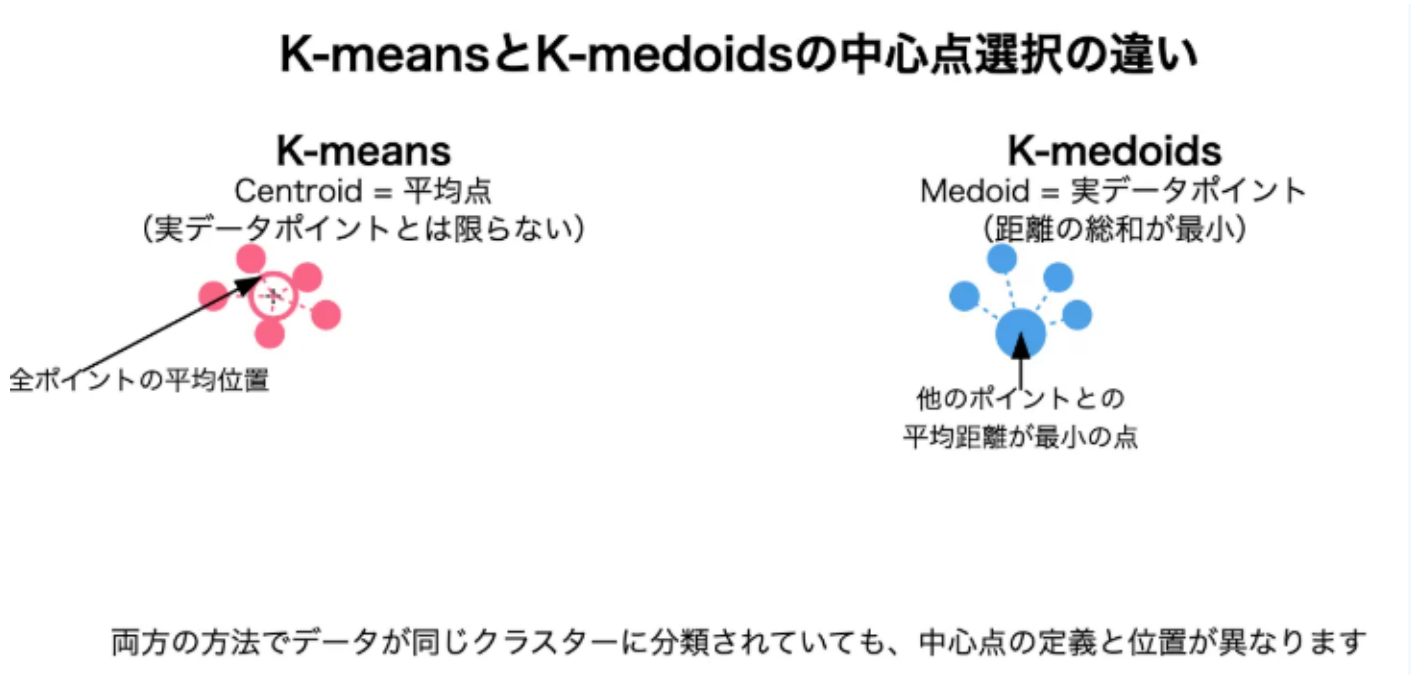


図 2: 画像 A

先行研究との比較

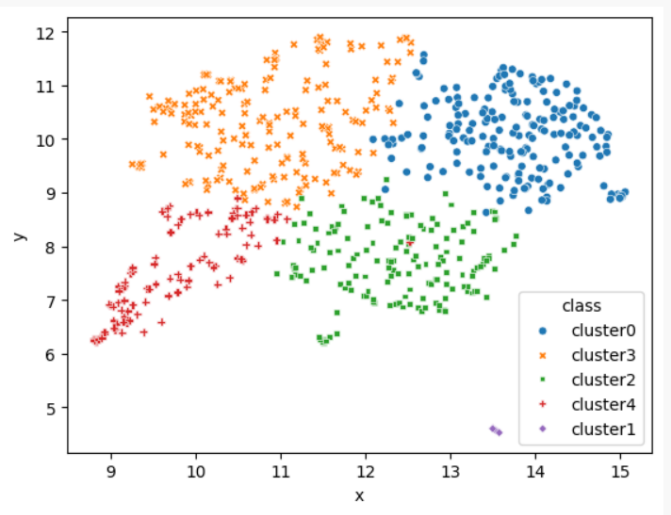


図 3: 平井さん

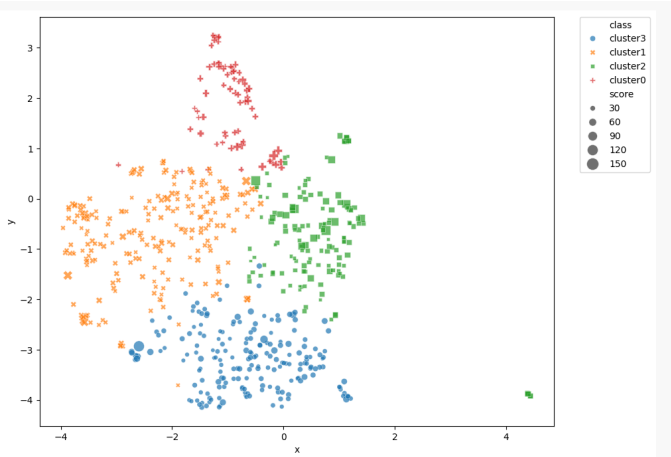


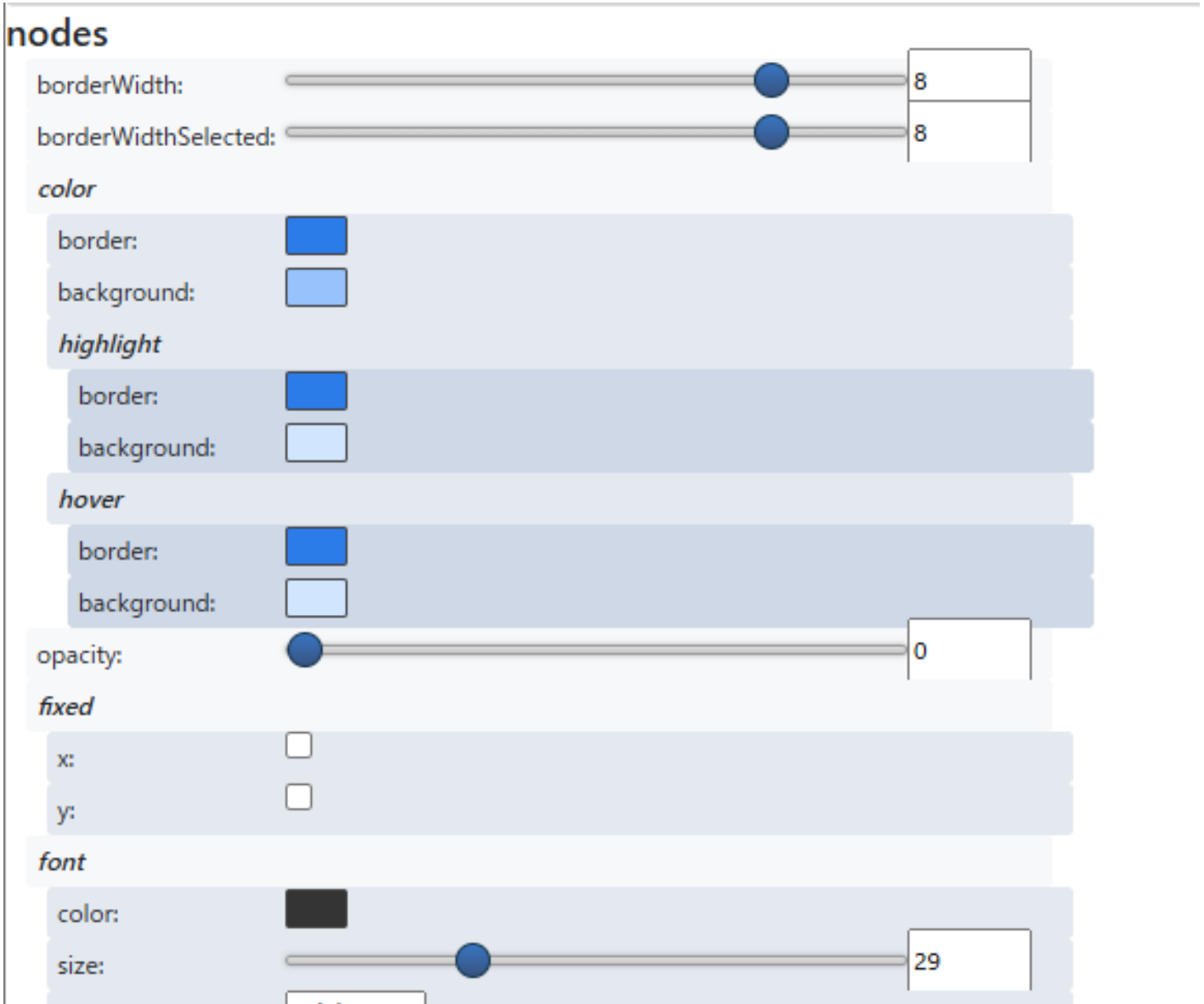
図 4: 今回の

追加したもの

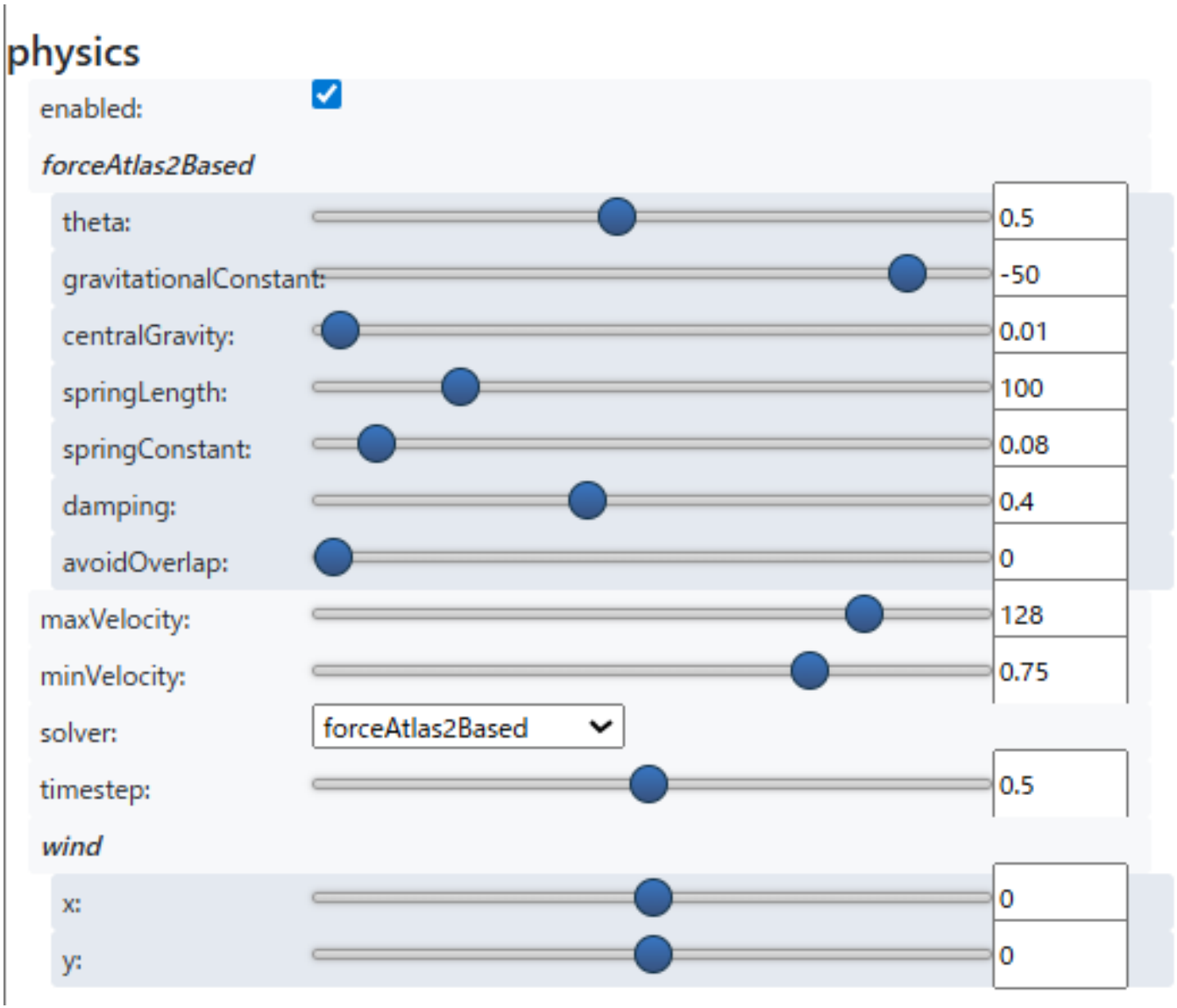
既存の 2D グラフに追加したもの

現在のシステムでは，すべての特許が「等しい点」として扱われている．これに「重要な特許」と「そうでない特許」を区別する機能を追加.
特許の重要度を定量化し, 可視化の際にノードの大きさで表現.
データ収集部に以下の指標を追加取得させる
被引用数: 他の特許からどれだけ引用されているか（基本特許ほど多い）.
請求項の数: 権利範囲の広さの指標.
ファミリ数: 何カ国に出願しているか（コストを掛けている＝重要度が高い）.
これらを重み付けして「重要度スコア」を算出.
メリット: ユーザーはマップを見た瞬間に「このクラスターの中心にある重要な特許はどれか」を一目で把握できるようになり，探索効率が向上する.

追加したもの


nodes
borderWidth:
8
borderWidthSelected:
8
color
border:
background:
highlight
border:
background:
hover
border:
background:
opacity:
0
fixed
x:
y:
font
color:
size:
29

追加したもの


physics
enabled:
forceAtlas2Based
theta:
0.5
gravitationalConstant:
-50
centralGravity:
0.01
springLength:
100
springConstant:
0.08
damping:
0.4
avoidOverlap:
0
maxVelocity:
128
minVelocity:
0.75
solver:
forceAtlas2Based
timestep:
0.5
wind
x:
0
y:
0

追加したもの

既存の 2D グラフに追加したもの

Nodes（ノード / 点の設定）ここでは，特許や単語を表す「点（ノード）」の見た目を設定.
borderWidth / borderWidthSelected:
点の枠線の太さ．通常時と，クリックして選択した時の太さを指定.
color (border, background, highlight, hover):
点の色設定. 通常時, 選択時（highlight）, マウスを乗せた時（hover）の枠線と背景色を決める.
opacity: 透明度です（0〜1）。
font (color, size, face, background...): 点に表示されるラベル（単語や特許番号）の文字設定. 文字色, サイズ, フォント（arial など）, 背景色などを変更できる.

追加したもの

追加したもの

scaling (min, max): 重要，これが今回の「特許評価スコアリング」に直結する設定.
プログラムで設定したスコア（value）に基づいて，点がどれくらい大きくなるかの範囲を指定する.
min: スコアが低い点の最小サイズ.
max: スコアが高い点の最大サイズ.
shape: 点の形です（ellipse=楕円, circle=円, box=四角など）．現在は ellipse になっている.
size: スケーリングを使わない場合の基準サイズ.
Physics（物理演算 / 動きの設定）ここでは，ネットワーク図がどのように広がり，配置されるかの「物理法則」を設定する.
enabled: チェックを入れると物理演算が有効になり，ノードがうねうねと動いて最適な位置を探す．外すと動きが止まる.
solver: 計算アルゴリズムの種類．現在は forceAtlas2Based が選択されている．これは大規模なネットワークを見やすく広げるのによく使われるモデル.

追加したもの

追加したもの

forceAtlas2Based の詳細設定:gravitationalConstant (引力定数): ここではマイナス値（-50）になっている．これは「反発力」を意味し，ノード同士が磁石の N 極と N 極のように反発し合う．値を小さく（マイナスを大きく）すると，ノード同士がより離れようとする．
centralGravity (中心引力): グラフ全体が画面の中心に引っ張られる力．これが弱いとグラフが画面外へ飛んでいってしまうことがある．
springLength (バネの長さ): つながっているノード同士（共起関係にある単語同士）の距離の基準．
springConstant (バネ定数): つながりの強さ．高いと，つながっているノード同士がギュッと近くに引き寄せられる．
avoidOverlap: 1 に近づけるほど，ノード同士が重ならないように調整される．文字が読みづらい場合に有効．

追加したもの

どう活用するか？

この画面でスライダーを動かすと，リアルタイムでグラフが変化する．
見やすい配置を探す:「ごちゃごちゃしている」と思ったら，gravitationalConstant をもっとマイナスにする（左へ動かす）か，springLength を大きくしてみてると，全体が広がる．
スコアの差を強調する: 重要な特許をもっと目立たせたい場合は，1 枚目の scaling の min を小さく，max を大きくすると，大小のメリハリがつく．
設定を保存する: ブラウザ上で調整して「これが見やすい！」という設定が見つかったら，その一番下（画像には写っていませんが）に generate options というボタンを押すと設定コードが表示されるので，それを Python コード（appli4.py）に反映させると，最初からその綺麗な形で表示されるようになる．

実行時間短縮

案

詳細ページの取得を Selenium から requests に変更する.
Janome Tokenizer の初期化をループの外に出す.

先行研究との比較

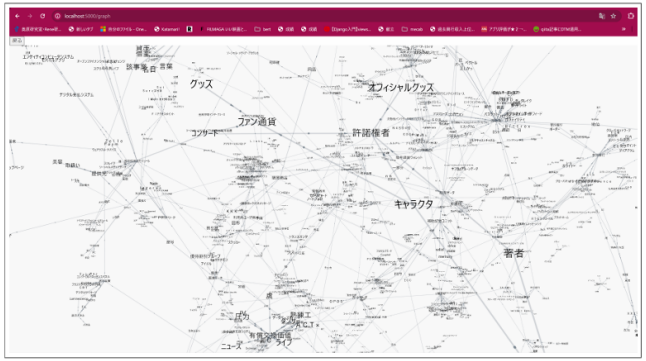


図 5.3: 出力された 3D グラフ

図 7: 平井さん

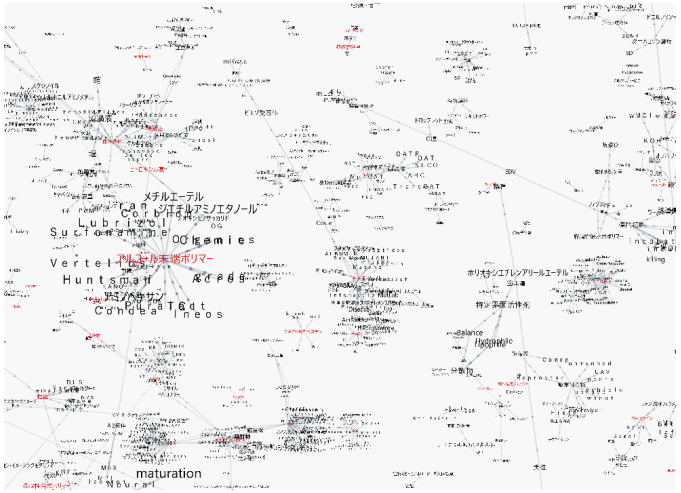


図 8: 今回の