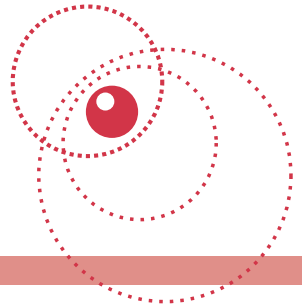


Web mining

Information Retrieval

9 October 2014

Pierre Senellart

TELECOM ParisTech

How to index Web content so as to answer (keyword-based) queries efficiently?

Context: set of text documents

- d_1 The jaguar is a New World mammal of the Felidae family.
- d_2 Jaguar has designed four new engines.
- d_3 For Jaguar, Atari was keen to use a 68K family device.
- d_4 The Jacksonville Jaguars are a professional US football team.
- d_5 Mac OS X Jaguar is available at a price of US $199 for Apple's new "family pack".
- d_6 One such ruling family to incorporate the jaguar into their name is Jaguar Paw.
- d_7 It is a big cat.

Initial text preprocessing steps

- Number of optional steps
- Highly depends on the application
- Highly depends on the document language (illustrated with English)

How to find the language used in a document?

- Meta-information about the document: often not reliable!
- Unambiguous scripts or letters: not very common!

한글

カタカナ

ⵜⵉⴼⵉⵏⴰⵖ

Għarbi

þorn

How to find the language used in a document?

- Meta-information about the document: often not reliable!
- Unambiguous scripts or letters: not very common!

  한글
  カタカナ
  ދިވެހި
  Għarbi
  Þorn

  Respectively: Korean Hangul, Japanese Katakana, Maldivian Dhivehi, Maltese, Icelandic
- Extension of this: frequent characters, or, better, frequent *k*-grams
- Use standard machine learning techniques (classifiers)

Separate text into tokens (words)

Not so easy!

- In some languages (Chinese, Japanese), words not separated by whitespace
- Deal consistently with acronyms, elisions, numbers, units, URLs, emails, etc.
- Compound words: *hostname*, *host-name* and *host name*. Break into two tokens or regroup them as one token? In any case, lexicon and linguistic analysis needed! Even more so in other languages as German.

Usually, remove punctuation and normalize case at this point

d_1 the$_1$ jaguar$_2$ is$_3$ a$_4$ new$_5$ world$_6$ mammal$_7$ of$_8$ the$_9$ felidae$_{10}$ family$_{11}$

d_2 jaguar$_1$ has$_2$ designed$_3$ four$_4$ new$_5$ engines$_6$

d_3 for$_1$ jaguar$_2$ atari$_3$ was$_4$ keen$_5$ to$_6$ use$_7$ a$_8$ 68k$_9$ family$_{10}$ device$_{11}$

d_4 the$_1$ jacksonville$_2$ jaguars$_3$ are$_4$ a$_5$ professional$_6$ us$_7$ football$_8$ team$_9$

d_5 mac$_1$ os$_2$ x$_3$ jaguar$_4$ is$_5$ available$_6$ at$_7$ a$_8$ price$_9$ of$_{10}$ us$_{11}$ \$199$_{12}$ for$_{13}$ apple's$_{14}$ new$_{15}$ family$_{16}$ pack$_{17}$

d_6 one$_1$ such$_2$ ruling$_3$ family$_4$ to$_5$ incorporate$_6$ the$_7$ jaguar$_8$ into$_9$ their$_{10}$ name$_{11}$ is$_{12}$ jaguar$_{13}$ paw$_{14}$

d_7 it$_1$ is$_2$ a$_3$ big$_4$ cat$_5$

Principle

Merge different forms of the same word, or of closely related words, into a single stem

- Not in all applications!
- Useful for retrieving documents containing *geese* when searching for *goose*
- Various degrees of stemming
- Possibility of building different indexes, with different stemming

Morphological stemming (lemmatization).

- Remove bound morphemes from words:
  - plural markers
  - gender markers
  - tense or mood inflections
  - etc.
- Can be linguistically very complex, cf:

  Les poules du couvent couvent.

  [The hens of the monastery brood.]
- In English, somewhat easy:
  - Remove final -s, -'s, -ed, -ing, -er, -est
  - Take care of semiregular forms (e.g., -y/-ies)
  - Take care of irregular forms (mouse/mice)
- But still some ambiguities: cf rose

Lemmatisation/Stemming.

- Merge lexically related terms of various parts of speech, such as *policy*, *politics*, *political* or *politician*
- For English, Porter's stemming [Porter, 1980]; stem *university* and *universal* to *univers*: not perfect!
- Possibility of coupling this with lexicons to merge (near-)synonyms

Phonetic stemming.

- Merge phonetically related words: search proper names with different spellings!
- For English, Soundex [US National Archives and Records Administration, 2007] stems *Robert* and *Rupert* to *R163*. Very coarse!

d_1 the$_1$ jaguar$_2$ be$_3$ a$_4$ new$_5$ world$_6$ mammal$_7$ of$_8$ the$_9$ felidae$_{10}$ family$_{11}$

d_2 jaguar$_1$ have$_2$ design$_3$ four$_4$ new$_5$ engine$_6$

d_3 for$_1$ jaguar$_2$ atari$_3$ be$_4$ keen$_5$ to$_6$ use$_7$ a$_8$ 68k$_9$ family$_{10}$ device$_{11}$

d_4 the$_1$ jacksonville$_2$ jaguar$_3$ be$_4$ a$_5$ professional$_6$ us$_7$ football$_8$ team$_9$

d_5 mac$_1$ os$_2$ x$_3$ jaguar$_4$ be$_5$ available$_6$ at$_7$ a$_8$ price$_9$ of$_{10}$ us$_{11}$ \$199$_{12}$ for$_{13}$ apple$_{14}$ new$_{15}$ family$_{16}$ pack$_{17}$

d_6 one$_1$ such$_2$ rule$_3$ family$_4$ to$_5$ incorporate$_6$ the$_7$ jaguar$_8$ into$_9$ their$_{10}$ name$_{11}$ be$_{12}$ jaguar$_{13}$ paw$_{14}$

d_7 it$_1$ be$_2$ a$_3$ big$_4$ cat$_5$

Principle

Remove uninformative words from documents, in particular to lower the cost of storing the index

determiners: *a*, *the*, *this*, etc.

function verbs: *be*, *have*, *make*, etc.

conjunctions: *that*, *and*, etc.

etc.

d_1 jaguar$_2$ new$_5$ world$_6$ mammal$_7$ felidae$_{10}$ family$_{11}$

d_2 jaguar$_1$ design$_3$ four$_4$ new$_5$ engine$_6$

d_3 jaguar$_2$ atari$_3$ keen$_5$ 68k$_9$ family$_{10}$ device$_{11}$

d_4 jacksonville$_2$ jaguar$_3$ professional$_6$ us$_7$ football$_8$ team$_9$

d_5 mac$_1$ os$_2$ x$_3$ jaguar$_4$ available$_6$ price$_9$ us$_{11}$ \$199$_{12}$ apple$_{14}$ new$_{15}$ family$_{16}$ pack$_{17}$

d_6 one$_1$ such$_2$ rule$_3$ family$_4$ incorporate$_6$ jaguar$_8$ their$_{10}$ name$_{11}$ jaguar$_{13}$ paw$_{14}$

d_7 big$_4$ cat$_5$

After all preprocessing, construction of an inverted index:

- Index of all terms, with the list of documents where this term occurs
- Small scale: disk storage, with memory mapping (cf. `mmap`) techniques; secondary index for offset of each term in main index
- Large scale: distributed on a cluster of machines; hashing gives the machine responsible for a given term
- Updating the index costly, so only batch operations (not one-by-one addition of term occurrences)

family	d_1, d_3, d_5, d_6
football	d_4
jaguar	$d_1, d_2, d_3, d_4, d_5, d_6$
new	d_1, d_2, d_5
rule	d_6
us	d_4, d_5
world	d_1

...

- Phrase queries, NEAR operator: need to keep position information in the index
- just add it in the document list!

family	$d_1/11$, $d_3/10$, $d_5/16$, $d_6/4$
football	$d_4/8$
jaguar	$d_1/2$, $d_2/1$, $d_3/2$, $d_4/3$, $d_5/4$, $d_6/8 + 13$
new	$d_1/5$, $d_2/5$, $d_5/15$
rule	$d_6/3$
us	$d_4/7$, $d_5/11$
world	$d_1/6$
...	

- Some term occurrences have more weight than others:
  - Terms occurring frequently in a given document: more relevant
  - Terms occurring rarely in the document collection as a whole: more informative
- Add Term Frequency—Inverse Document Frequency weighting to occurrences;

$$\text{tfidf}(t, d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}} \cdot \log \frac{|D|}{|\{d' \in D \mid n_{t,d'} > 0\}|}$$

$n_{t,d}$	number of occurrences of t in d
D	set of all documents

- Store documents (along with weight) in decreasing weight order in the index

family	$d_1/11/.13$, $d_3/10/.13$, $d_6/4/.08$, $d_5/16/.07$
football	$d_4/8/.47$
jaguar	$d_1/2/.04$, $d_2/1/.04$, $d_3/2/.04$, $d_4/3/.04$, $d_6/8+13/.04$, $d_5/4/.$
new	$d_2/5/.24$, $d_1/5/.20$, $d_5/15/.10$
rule	$d_6/3/.28$
us	$d_4/7/.30$, $d_5/11/.15$
world	$d_1/6/.47$
...	

- Single keyword query: just consult the index and return the documents in index order.
- Boolean multi-keyword query

(*jaguar* AND *new* AND NOT *family*) OR *cat*

Same way! Retrieve document lists from all keywords and apply adequate set operations:

AND	intersection
OR	union
AND NOT	difference

- Global score: some function of the individual weight (e.g., addition for conjunctive queries)
- Position queries: consult the index, and filter by appropriate condition

$$t_1 \text{ AND } \ldots \text{ AND } t_n$$

$$t_1 \text{ OR } \ldots \text{ OR } t_n$$

Problem

Find the top-k results (for some given k) to the query, without retrieving all documents matching it.

Notations:

$s(t, d)$ weight of t in d (e.g., tfidf)

$g(s_1, \ldots, s_n)$ monotonous function that computes the global score (e.g., addition)

(with an additional direct index giving $s(t, d)$)

1. Let R be the empty list and $m = +\infty$.
2. For each $1 \leq i \leq n$:
   - 2.1 Retrieve the document $d^{(i)}$ containing term t_i that has the next largest $s(t_i, d^{(i)})$.
   - 2.2 Compute its global score $g_{d^{(i)}} = g(s(t_1, d^{(i)}), \ldots, s(t_n, d^{(i)}))$ by retrieving all $s(t_j, d^{(i)})$ with $j \neq i$.
   - 2.3 If R contains less than k documents, or if $g_{d^{(i)}}$ is greater than the minimum of the score of documents in R, add $d^{(i)}$ to R (and remove the worst element in R if it is full).
3. Let $m = g(s(t_1, d^{(1)}), s(t_2, d^{(2)}), \ldots, s(t_n, d^{(n)}))$.
4. If R contains k documents, and the minimum of the score of the documents in R is greater than or equal to m, return R.
5. Redo step 2.

- HTML: text + meta-information + structure
- Possibly: separate index for meta-information (title, keywords)
- Increase weight of structurally emphasized content in index
- Tree structure can also be queried with XPath or XQuery, but not very useful on the Web as a whole, because of tag soup and lack of consistency.

- Basic approach: index text from context of the media
  - surrounding text
  - text in or around the links pointing to the content
  - filenames
  - associated subtitles (hearing-impaired track on TV)
- Elaborate approach: index and search the media itself, with the help of speech recognition and sound, image, and video analysis. Mostly still experimental!
  - TrackID, Shazam: identify a song played on the radio
  - Musipedia: look for music by whistling a tune, `http://www.musipedia.org/`, `http://www.midomi.com/`
  - Image search from a similar image, `http://images.google.com/`, Google Goggles
  - Voxalead, `http://voxaleadnews.labs.exalead.com/`

Important pages are pages pointed to by important pages.

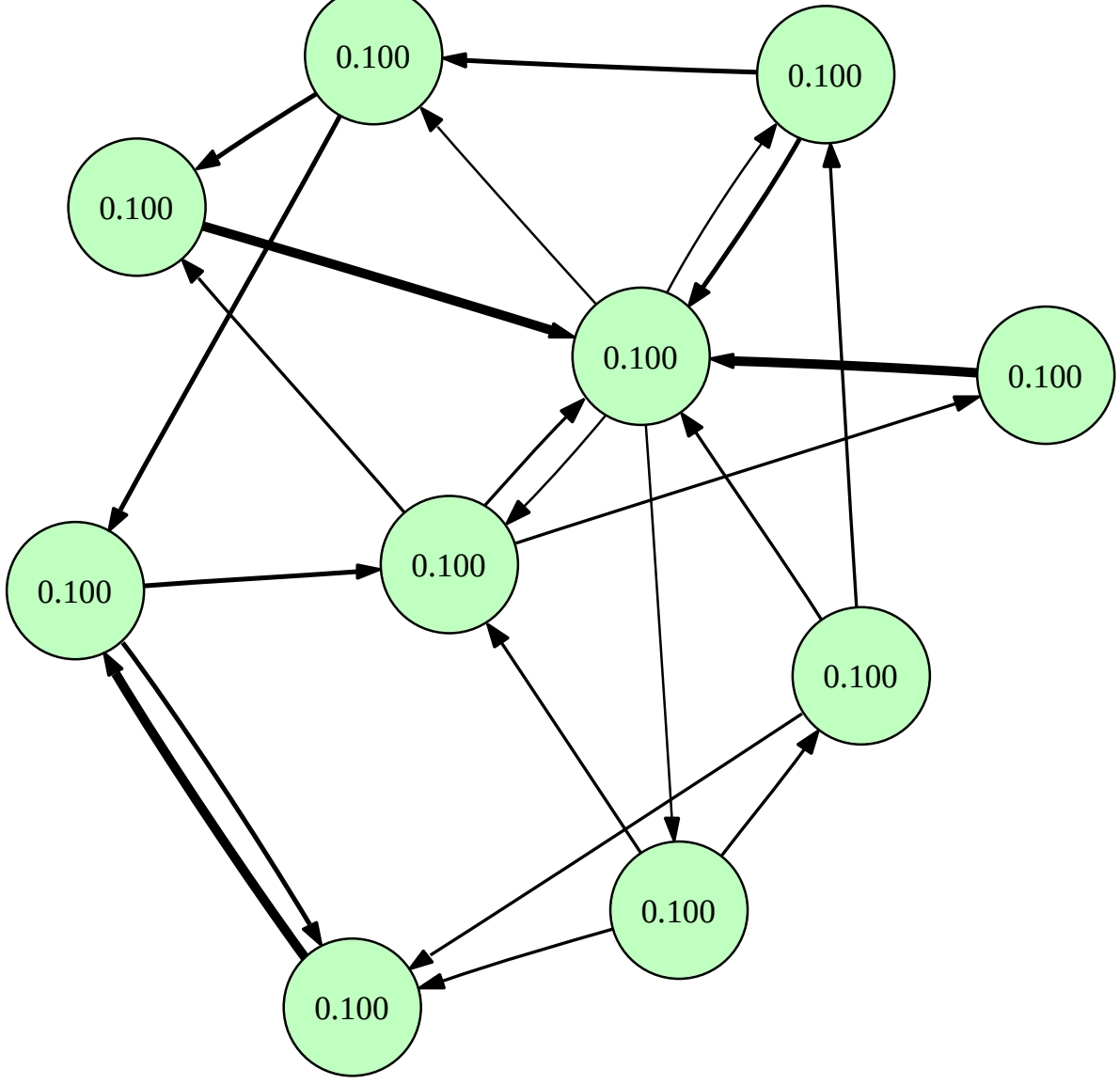
$$\begin{cases} g_{ij} = 0 & \text{if there is no link between page } i \text{ and } j; \\ g_{ij} = \frac{1}{n_i} & \text{otherwise, with } n_i \text{ the number of outgoing links of page } i. \end{cases}$$

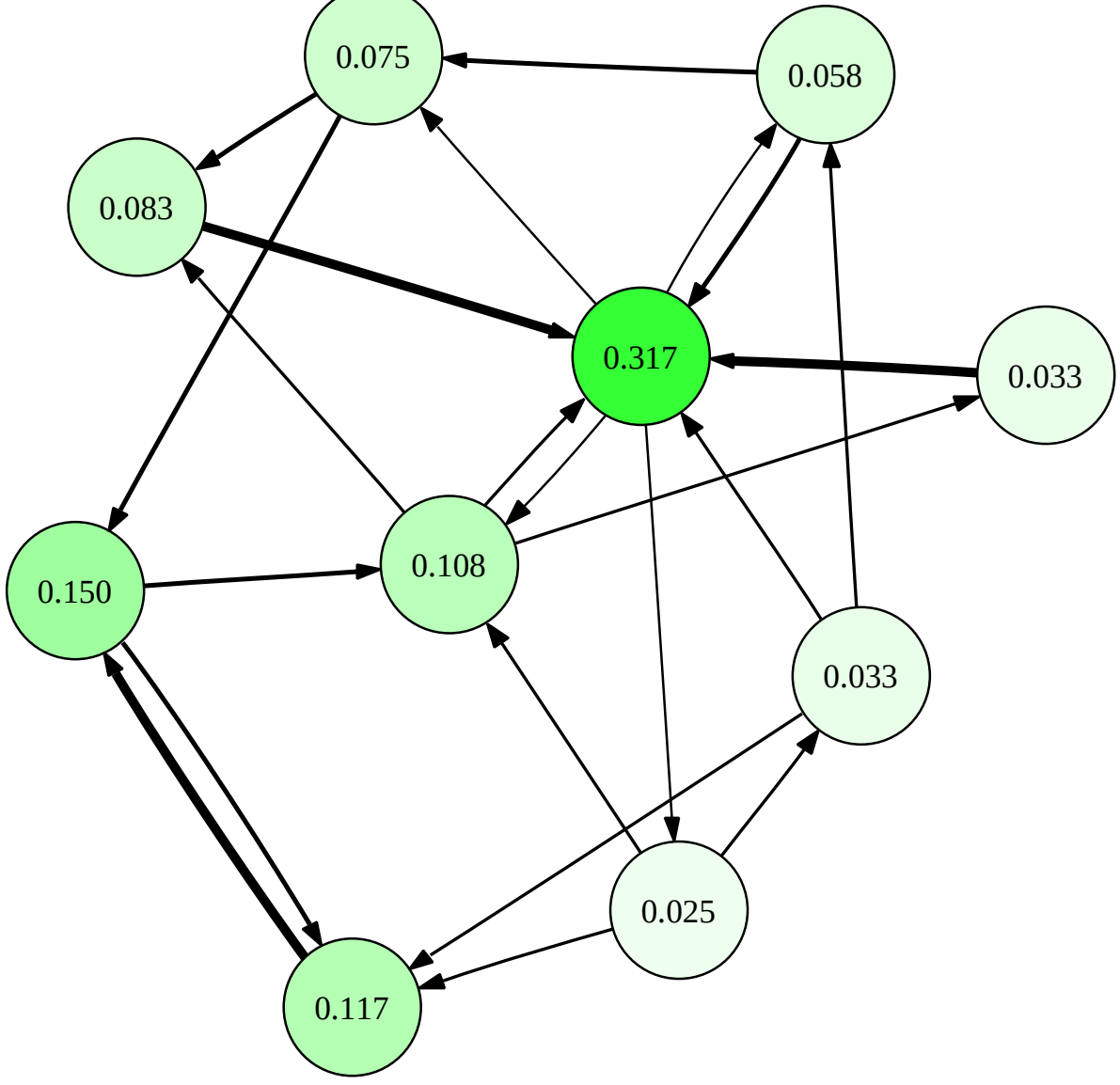
Definition (Tentative)

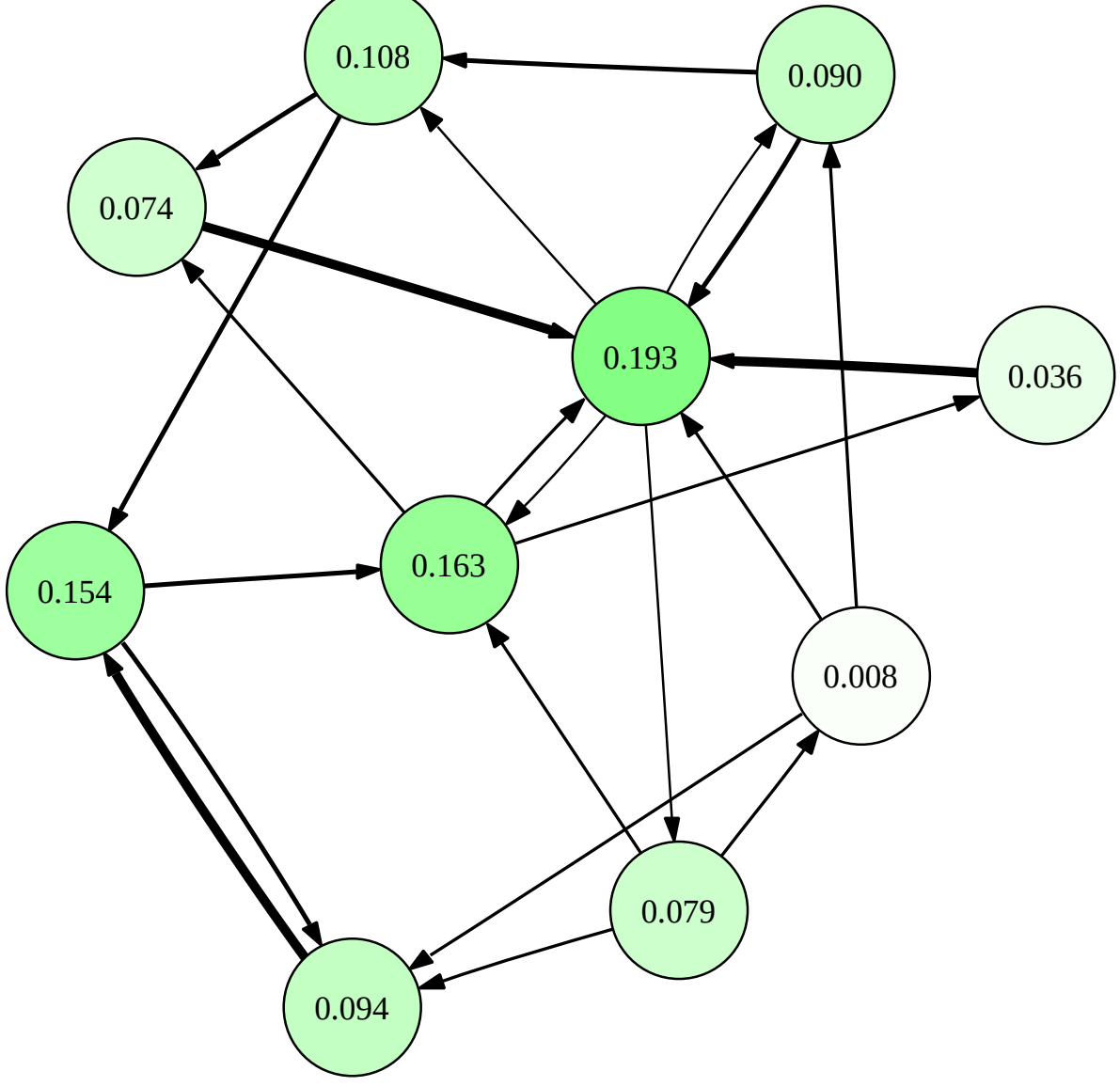
Probability that the surfer following the random walk in G has arrived on page i at some distant given point in the future.

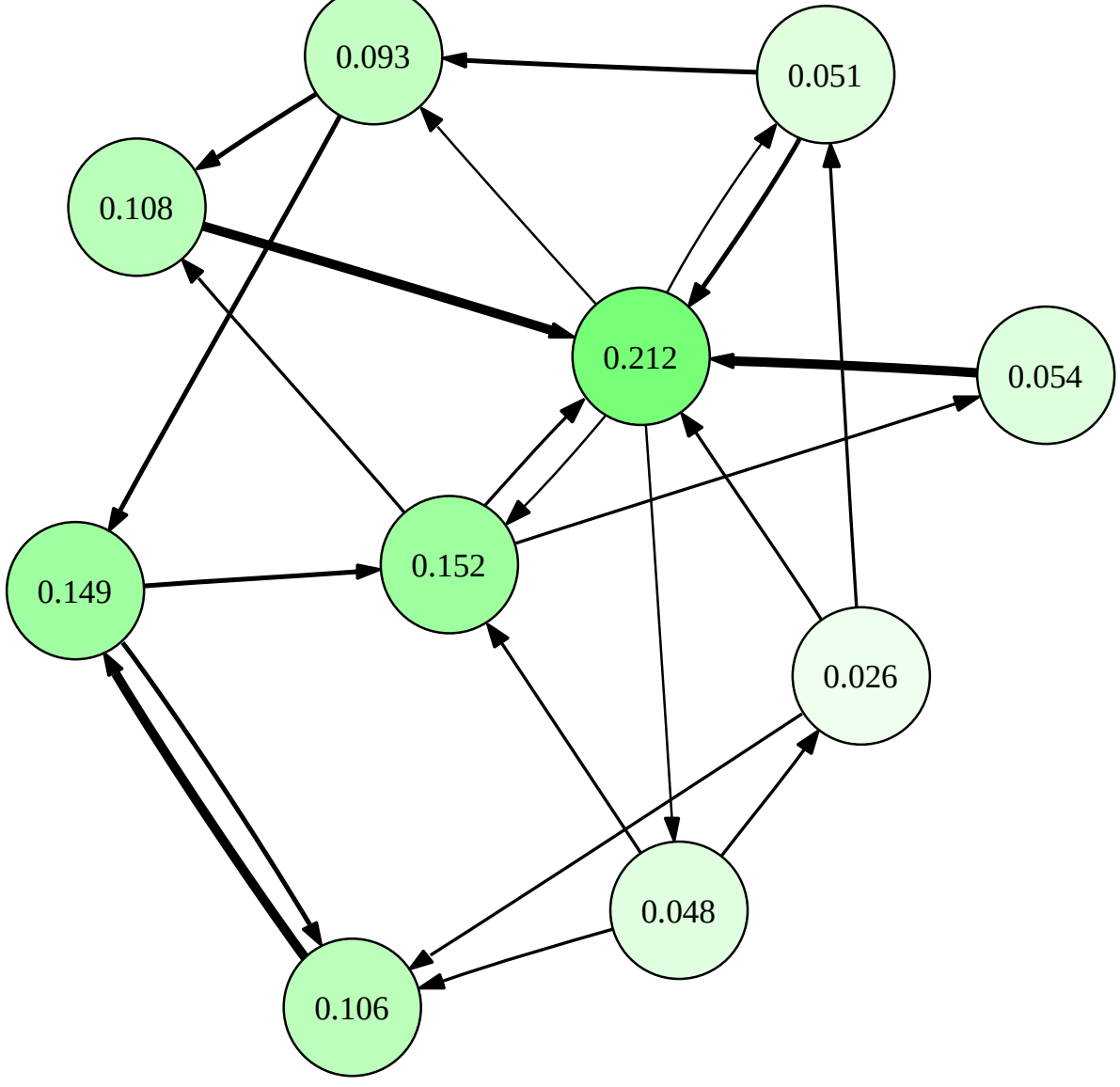
$$\mathrm{pr}(i) = \left(\lim_{k \to +\infty} (G^T)^k v \right)_i$$

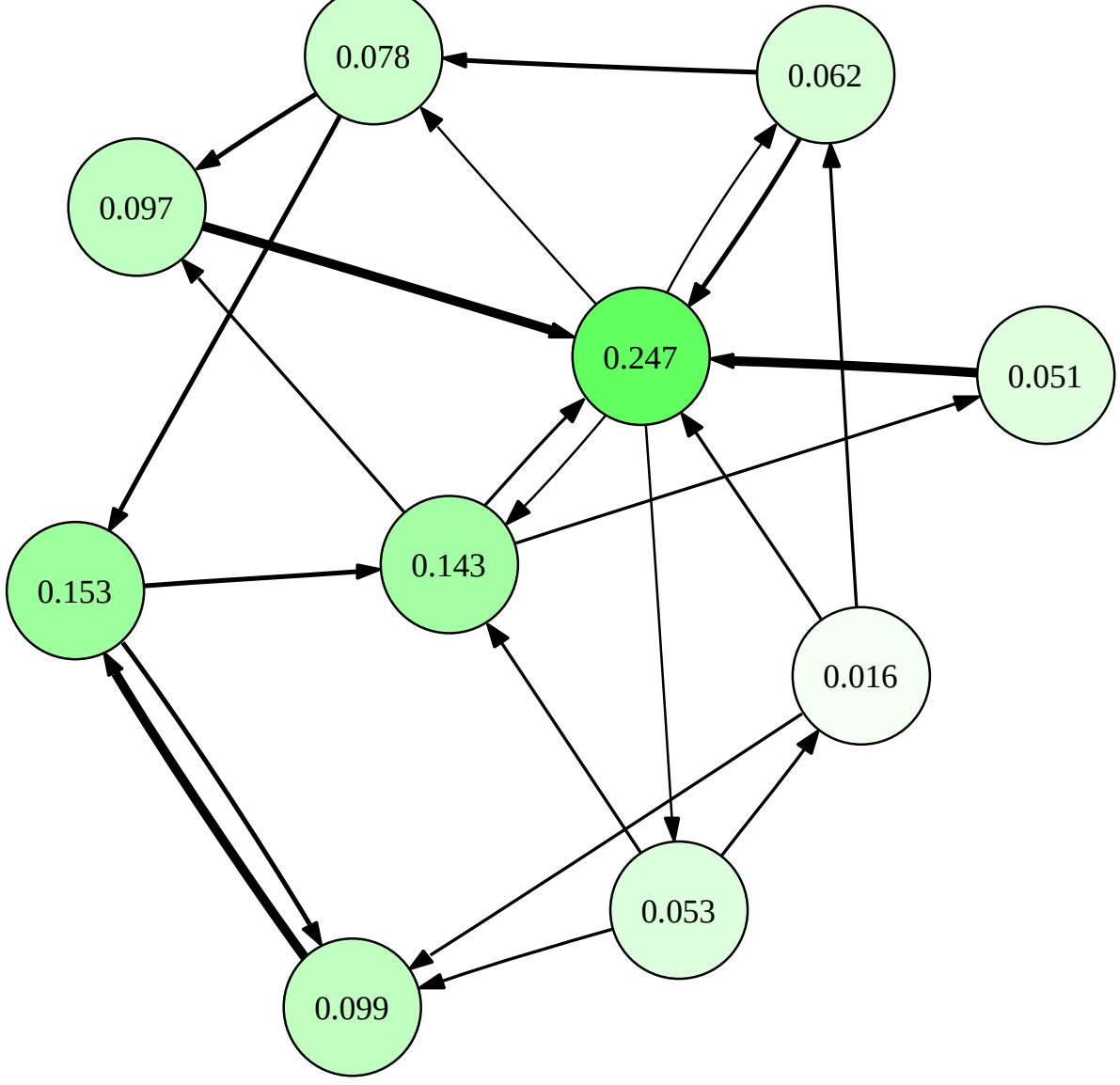
where v is some initial column vector.

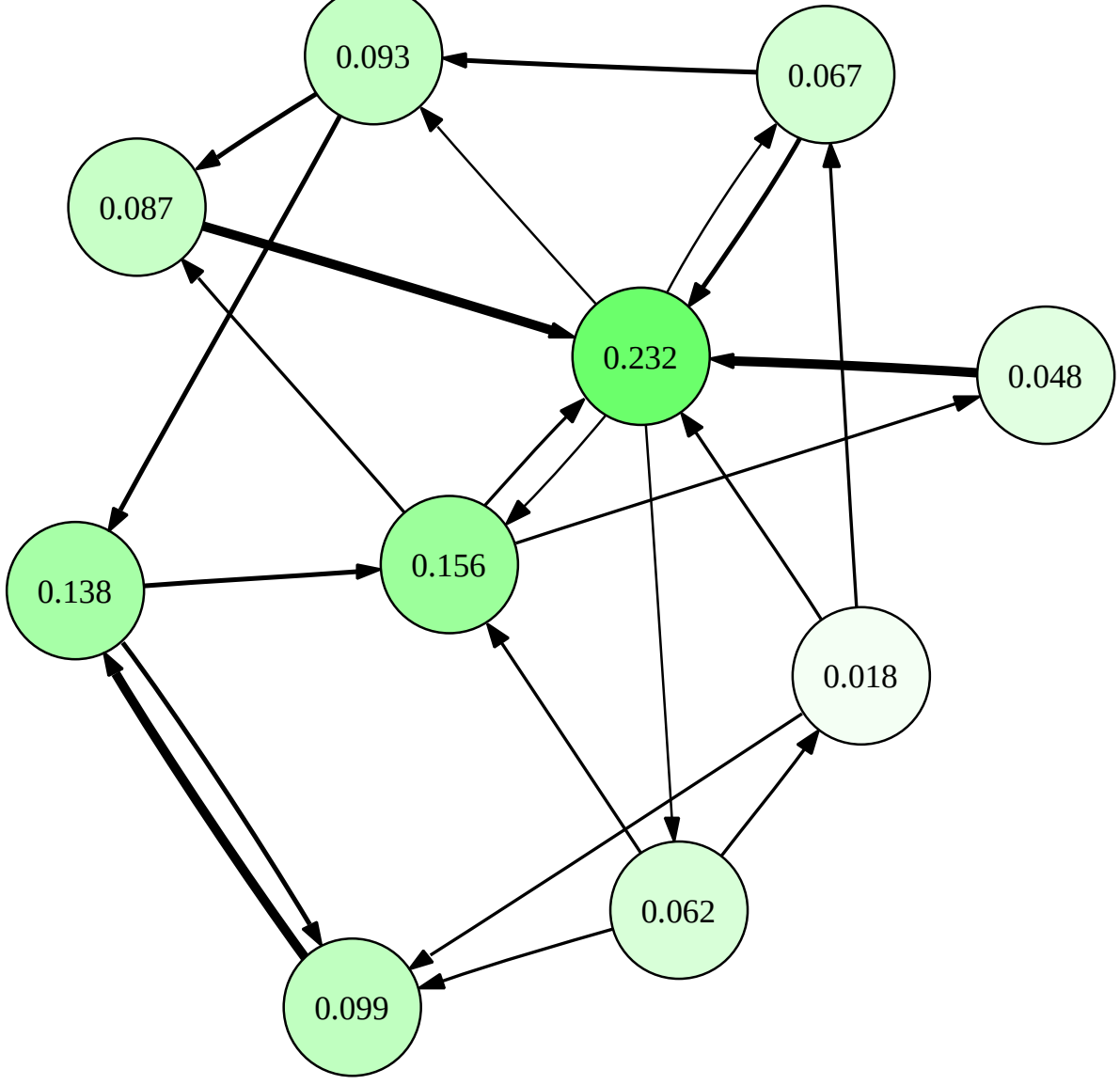


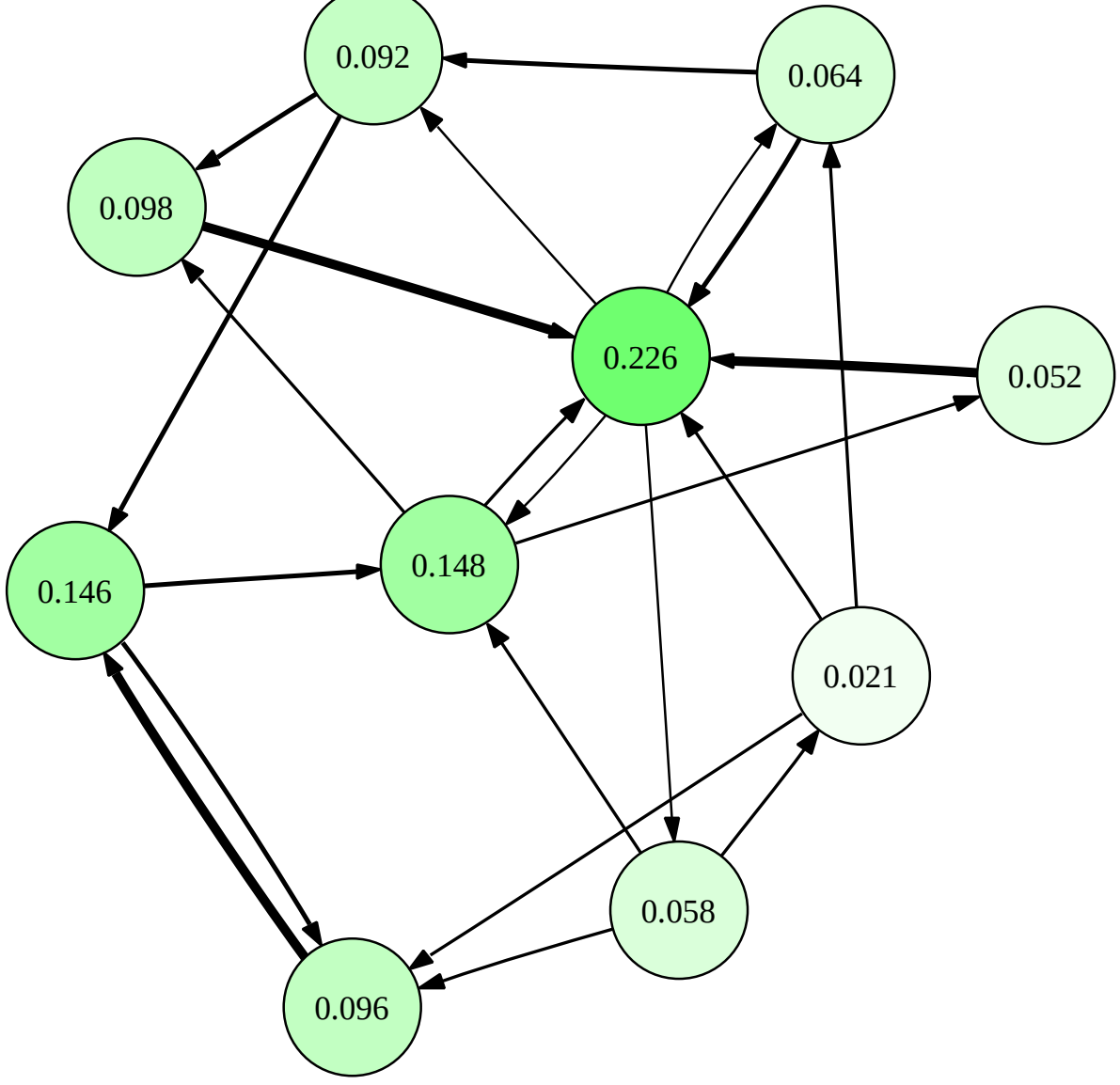


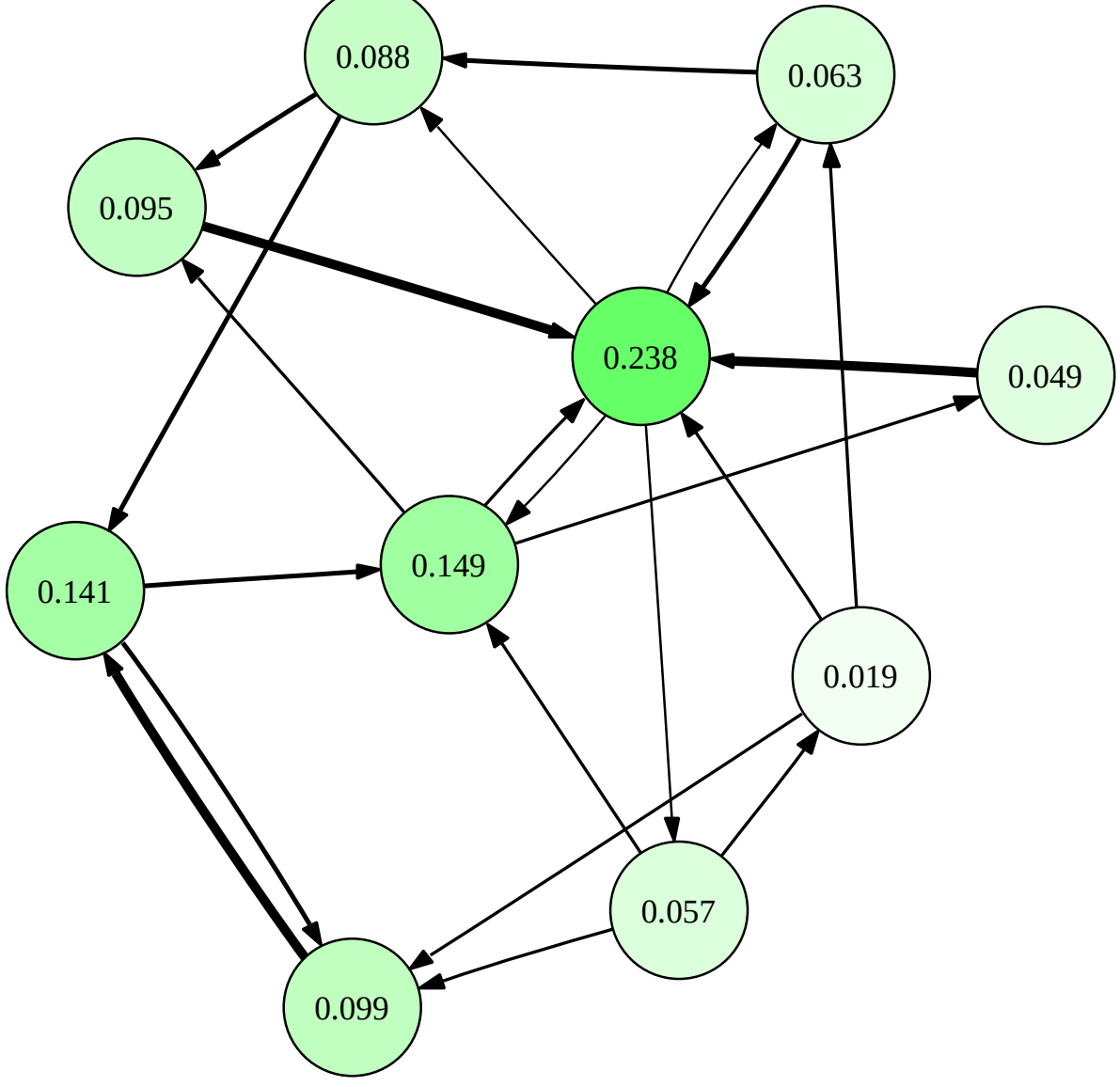


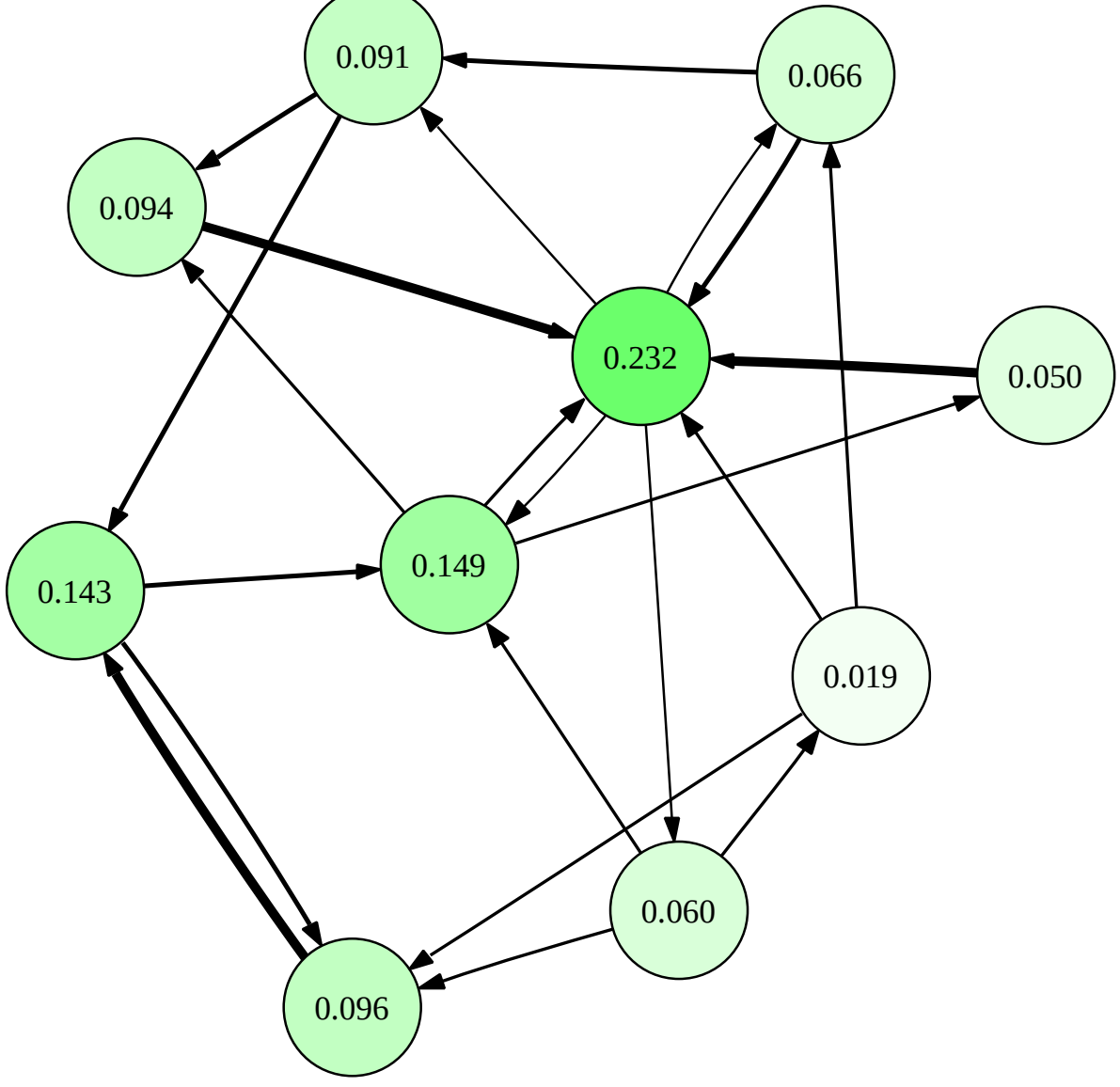


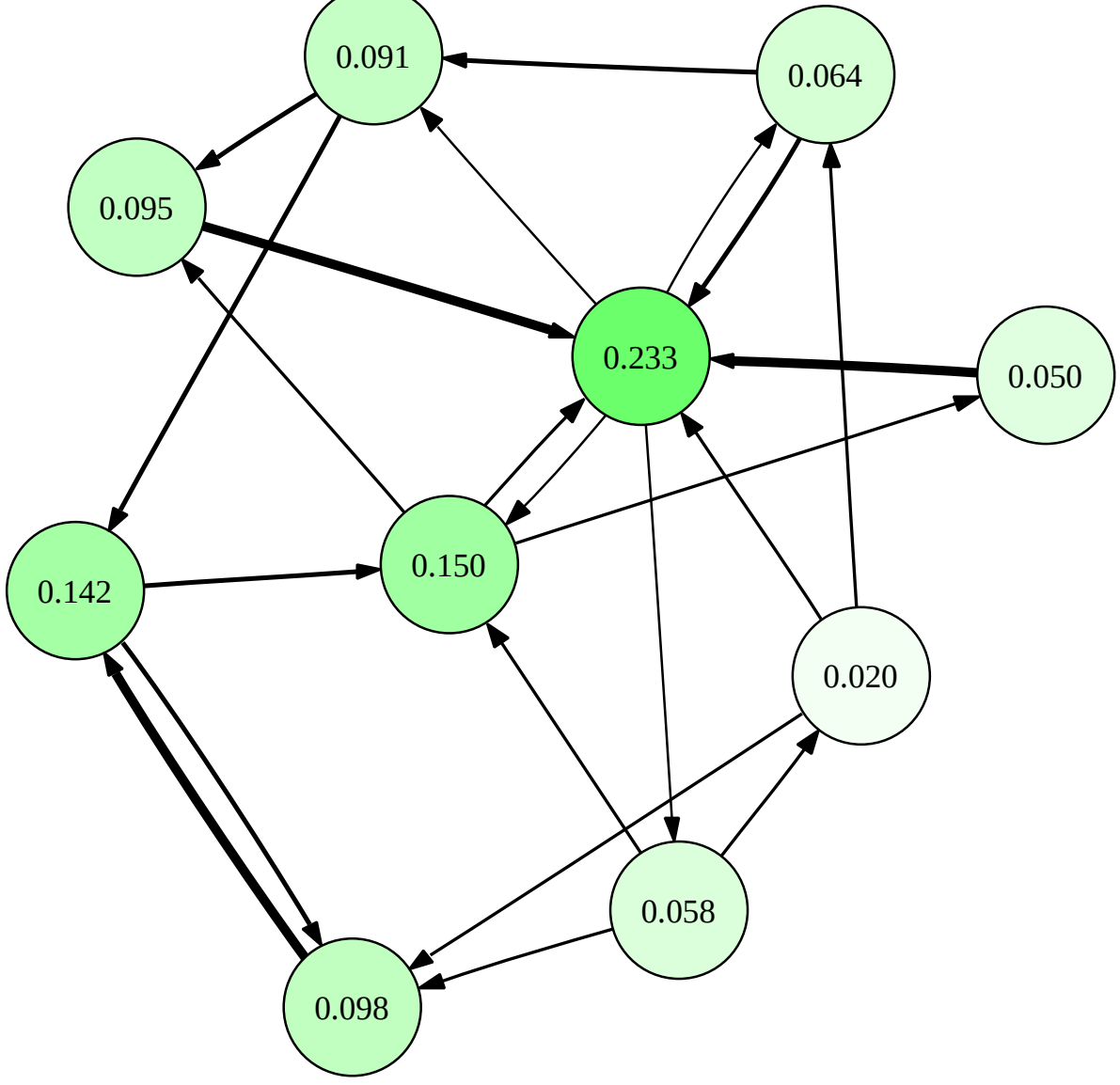


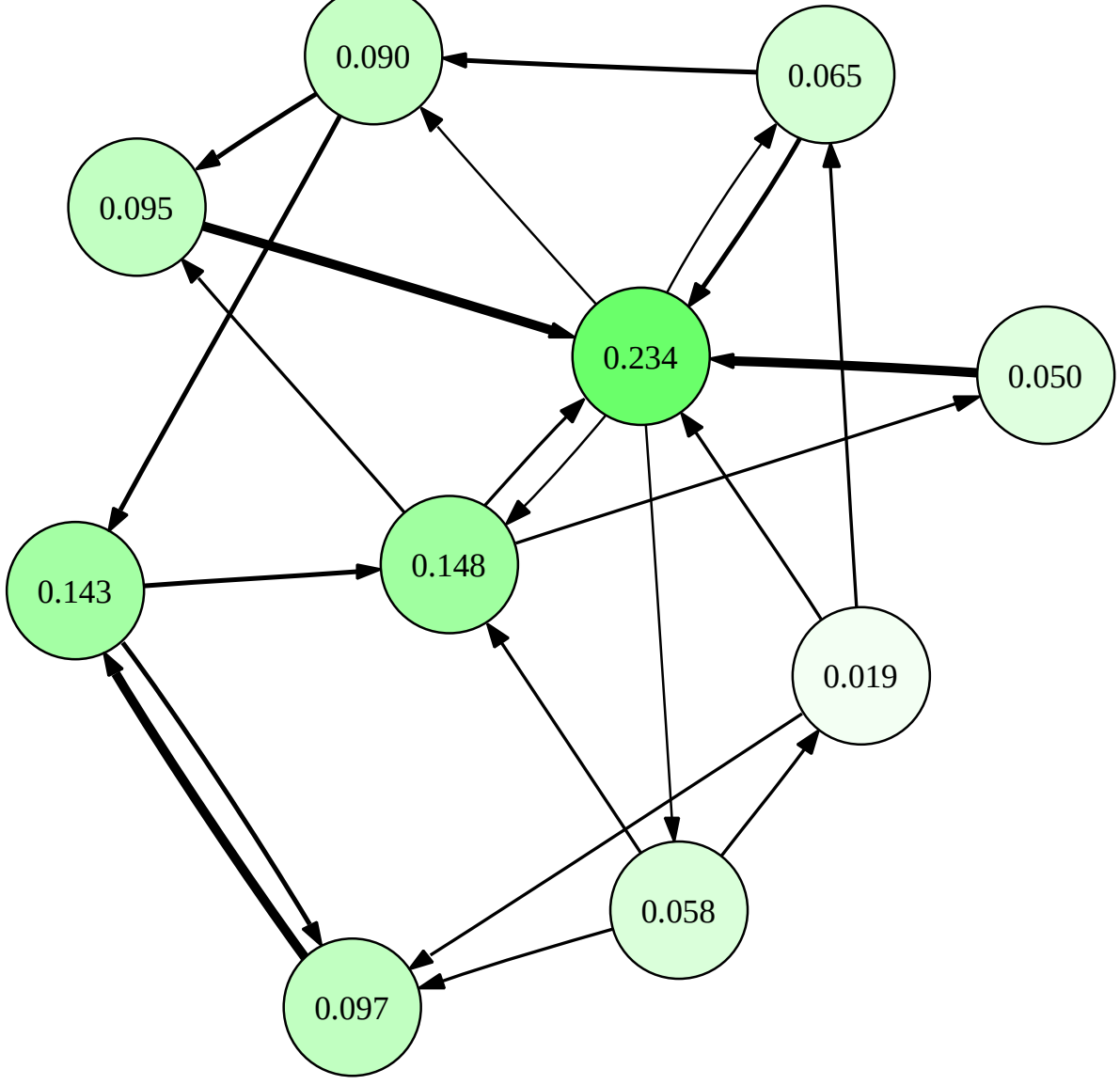


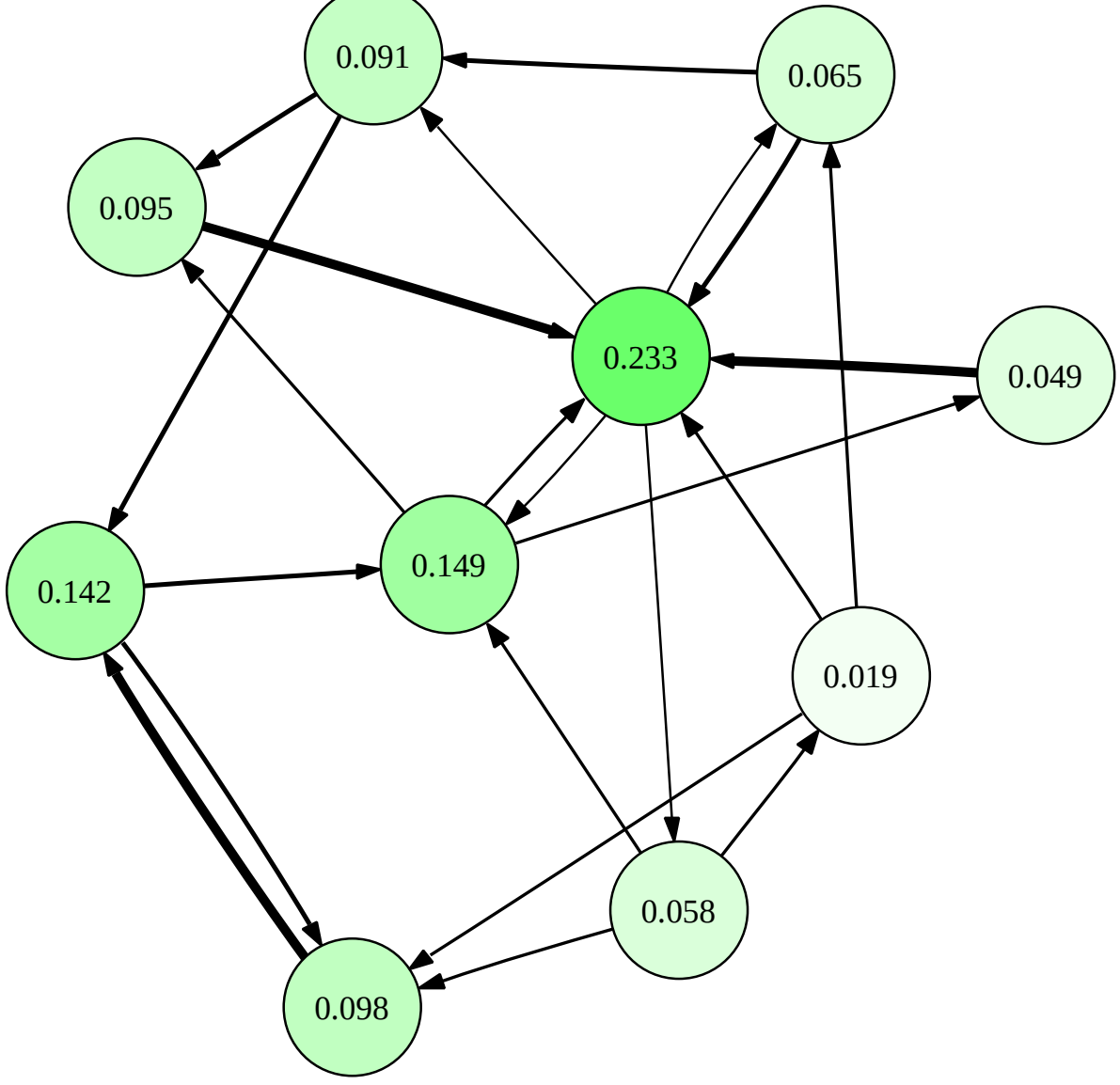


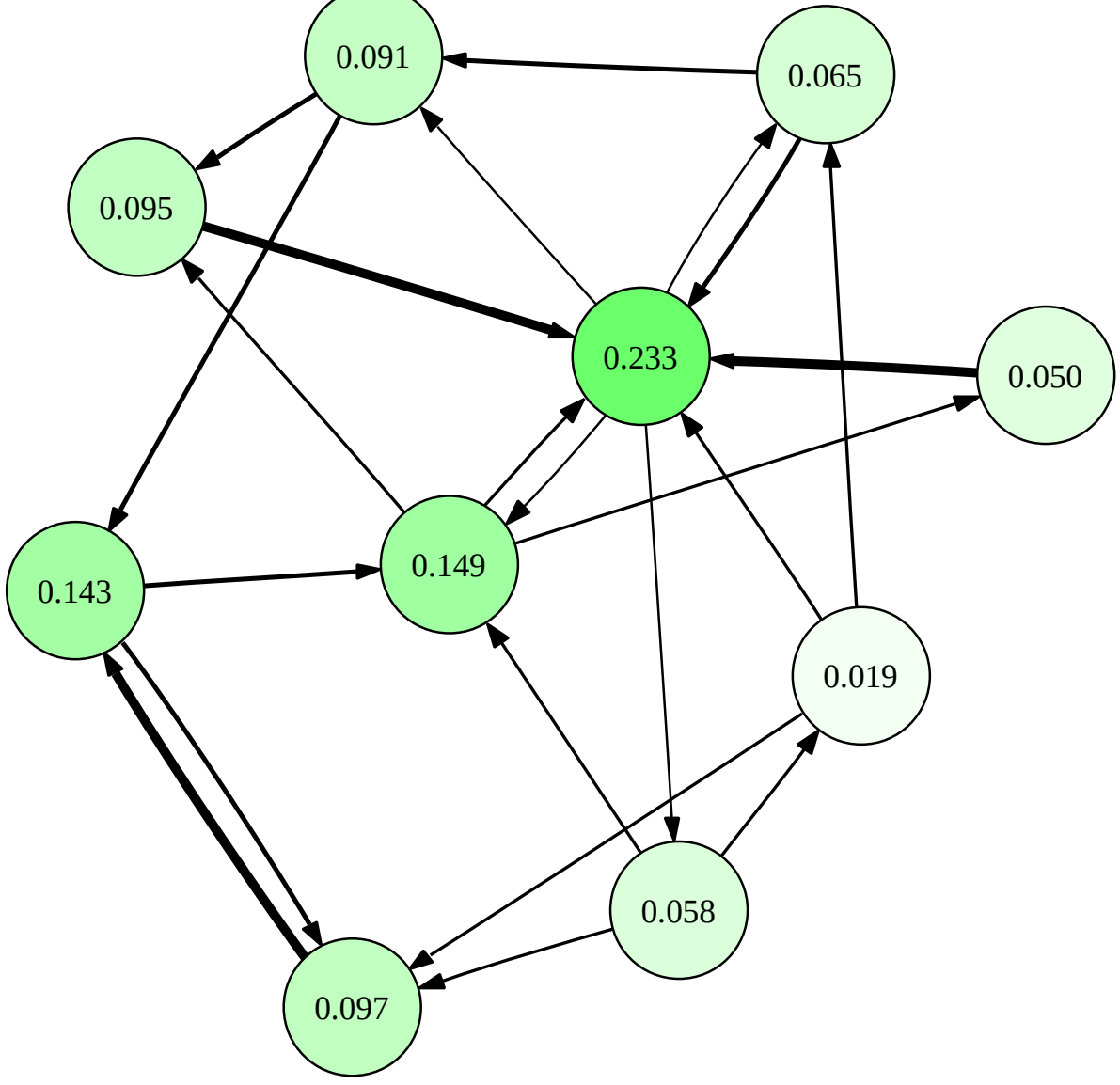


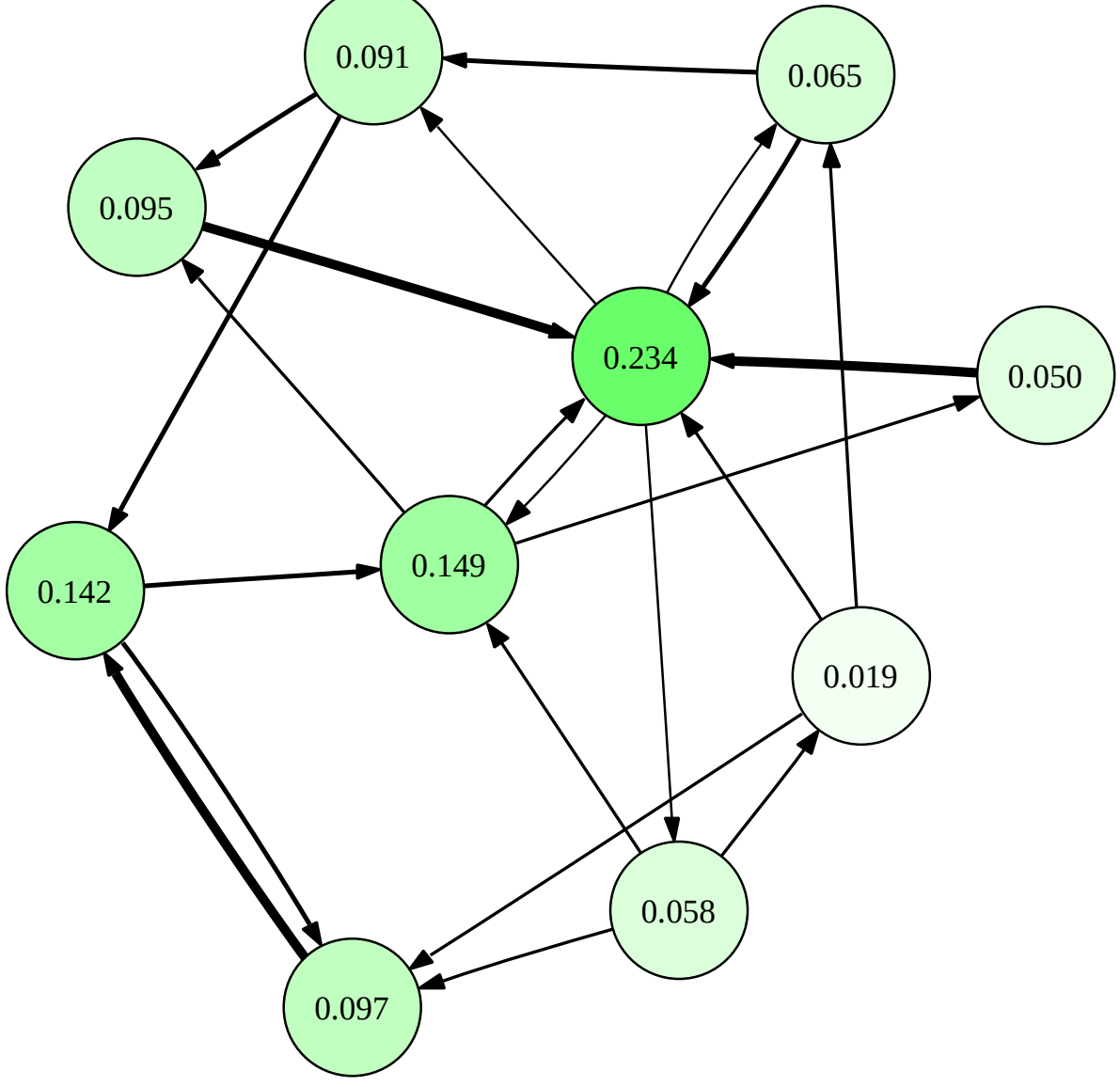












May not always converge, or convergence may not be unique.
To fix this, the random surfer can at each step randomly jump to any page of the Web with some probability d ($1 - d$: damping factor).

$$\mathrm{pr}(i) = \left(\lim_{k \to +\infty} ((1-d)G^T + dU)^k v \right)_i$$

where U is the matrix with all $\frac{1}{N}$ values with N the number of vertices.

- PageRank: global score, independent of the query
- Can be used to raise the weight of important pages:

$$\text{weight}(t, d) = \text{tfidf}(t, d) \times \text{pr}(d),$$

- This can be directly incorporated in the index.

Definition

Fraudulent techniques that are used by unscrupulous webmasters to artificially raise the visibility of their website to users of search engines

Purpose: attracting visitors to websites to make profit.

Unceasing war between spamdexers and search engines

Put unrelated terms in:

- meta-information (`<meta name="description">`, `<meta name="keywords">`)
- text content hidden to the user with JavaScript, CSS, or HTML presentational elements

Countertechnique

- Ignore meta-information
- Try and detect invisible text

Technique

Huge number of hosts on the Internet used for the sole purpose of referencing each other, without any content in themselves, to raise the importance of a given website or set of websites.

Countertechnique

- Detection of websites with empty or duplicate content
- Use of heuristics to discover subgraphs that look like link farms

Technique

Pollute user-editable websites (blogs, wikis) or exploit security bugs to add artificial links to websites, in order to raise its importance.

Countertechnique

`rel="nofollow"` attribute to `<a>` links not validated by a page's owner

- Faire attention à l'accessibilité aux robots des moteurs de recherche (Flash, JavaScript, etc.)
- Éventuellement prévoir des versions HTML pures alternatives
- Site accessible à une URL unique (par exemple, entre `http://www.toto.com/` et `http://toto.com/`, si les deux permettent d'accéder au site, l'un doit être une redirection (HTTP) vers l'autre)
- Structure cohérente (hébergement sur un seul serveur, contenu des URLs pertinentes), bonne structure de liens (de n'importe quelle page, il doit être possible d'atteindre la page principale en un clic, et n'importe quelle autre page en quelques clics)
- Liens vers les autres sites pertinents, que le site n'apparaisse pas complètement isolé

- Pas de tentatives de spamdexing: pas de texte invisible, pas [illegible] dans les mots-clefs de la balise `<meta>` (assez inutiles de toute façon), etc.
- Faire apparaître des liens vers le site (surtout vers la page principale, et éventuellement pages internes quand c'est pertinent) sur d'autres sites Web (référencement dans des annuaires, sur les pages Web des individus/sociétés en lien avec le contenu du site, etc.).
- Éventuellement s'il est difficile de faire ainsi, soumettre le site aux différents moteurs de recherche (p. ex., `http://www.google.com/addurl/` pour Google).
- Se méfier des sociétés proposant un meilleur référencement contre rémunération; leurs pratiques sont mal vues par les moteurs de recherche.
- Et enfin. . . avoir du contenu intéressant !

What you should remember

- The inverted index model, associated tools and techniques
  - Main ideas behind Fagin's TA and NRA
  - The document vector space model

Software

- Most DBMS have text indexing capabilities (e.g., MySQL's `FULLTEXT` indexes)
- Apache Lucene, free software library for information retrieval

To go further

- A good textbook [Manning et al., 2008]. Available online, along with slides!
- A very influential paper on top-k algorithms: [Fagin et al., 2001]

Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proc. PODS*, Santa Barbara, USA, May 2001.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich SchÃ¼tze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, United Kingdom, 2008. Available online at http://informationretrieval.org.

Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3): 130–137, July 1980.

US National Archives and Records Administration. The Soundex indexing system. `http://www.archives.gov/genealogy/census/soundex.html`, May 2007.