Topic Modeling

Orion Penner

31/30.05.2019

Competition and Innovation Summer School

Ulcinj, Montenegro

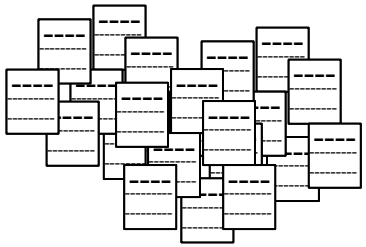
Overview

Overview

A **topic model** is a statistical model for extracting the abstract “topics” from a set of documents.

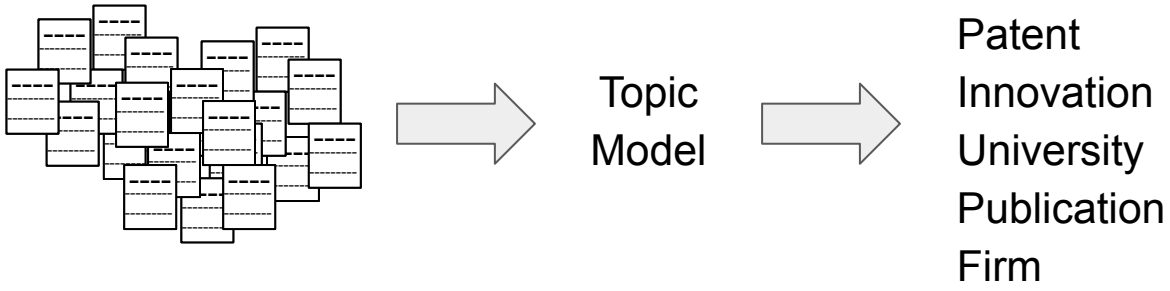
Overview

A **topic model** is a statistical model for extracting the abstract “topics” from a set of documents.



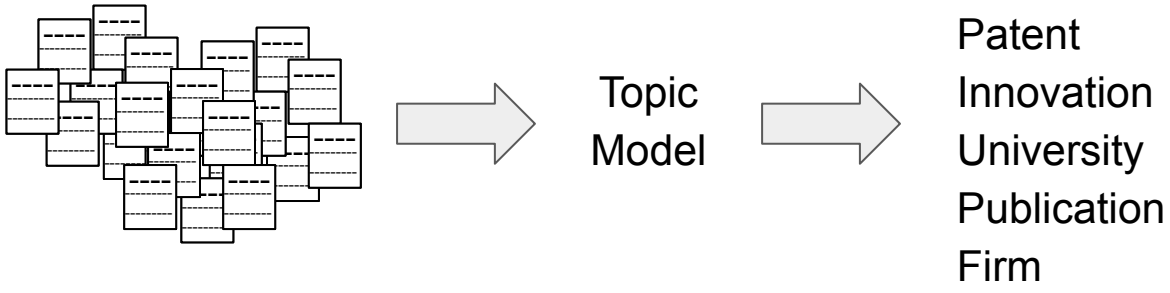
Overview

A **topic model** is a statistical model for extracting the abstract “topics” from a set of documents.



Overview

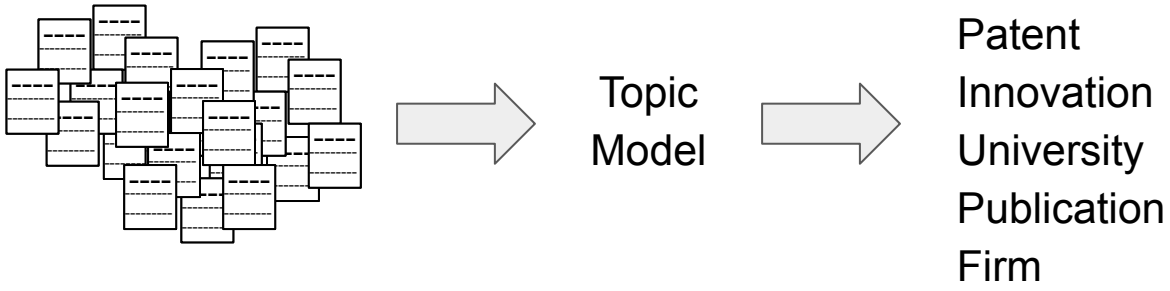
A **topic model** is a statistical model for extracting the abstract “topics” from a set of documents.



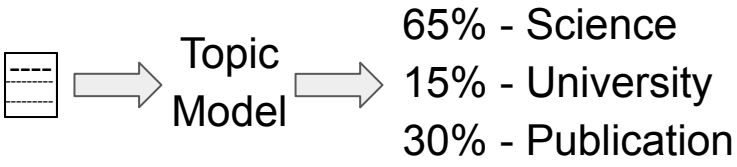
The more interesting feature is that they tell us the relative weight of each topic within each individual document.

Overview

A **topic model** is a statistical model for extracting the abstract “topics” from a set of documents.

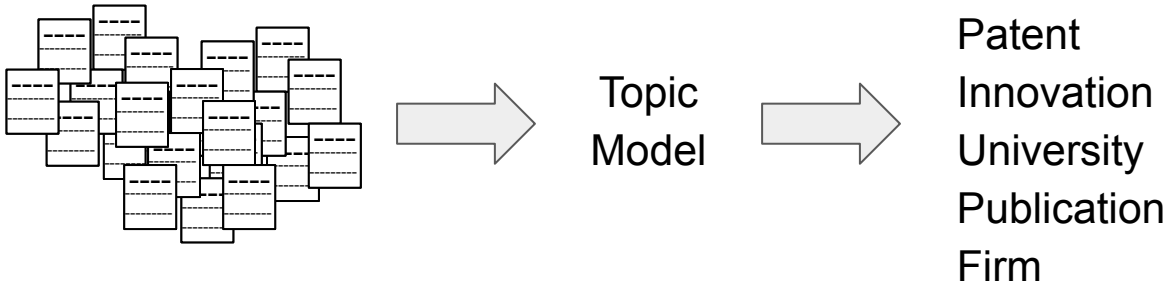


The more interesting feature is that they tell us the relative weight of each topic within each individual document.

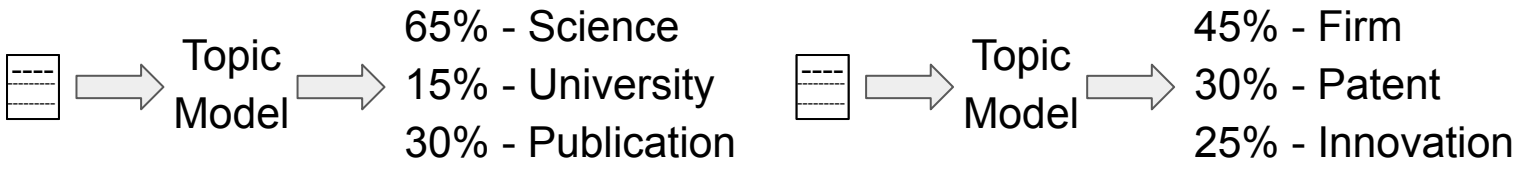


Overview

A **topic model** is a statistical model for extracting the abstract "topics" from a set of documents.

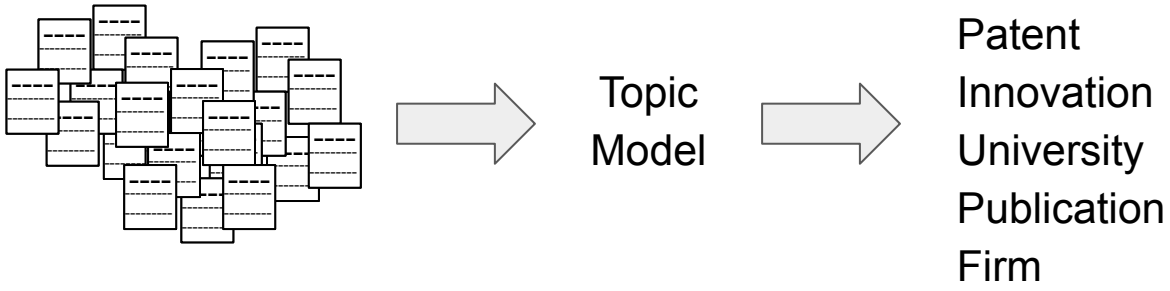


The more interesting feature is that they tell us the relative weight of each topic within each individual document.

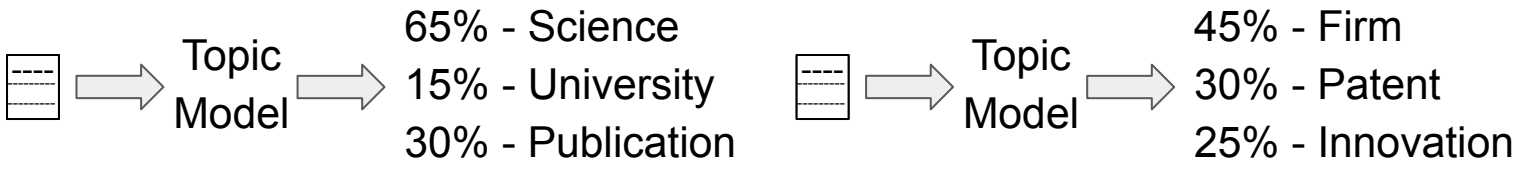


Overview

A **topic model** is a statistical model for extracting the abstract "topics" from a set of documents.



The more interesting feature is that they tell us the relative weight of each topic within each individual document.



But the reality is, of course, a lot messier than this!

Roadmap

Introduction
- Some facts and history
- What can you do with them?
- How it works – Conceptually and in practice

Two common topic models
- Non-Negative Matrix Factorization (NMF)
- word2vec & doc2vec

Using Topic Models
- Common packages
- Evaluating Robustness

Some facts and history

Topic models are a class of statistical models aimed at extracting the abstract “topics” of a set of documents.

Typically they span the fields of Machine Learning (ML) and Natural Language Processing (NLP). Although a few may fall into just dimensional reduction.

Most topic models are *unsupervised*.

On the whole this is a relatively new area of research – earliest NLP based topic models were published in the late 90’s.

But similar approaches designed for information retrieval do go back a bit farther (80’s) *ex*. Latent Semantic Analysis (LSA) and Latent Semantic Indexing (LSI).

What can you do with them?

A good way of looking at it is:

> What would you do with your favorite text based data if you could extract the ideas within each document at an arbitrary level of resolution?

First I'll recap one from before.

And then cover two things I currently have ongoing.

FOMC

Transparency and Deliberation Within the FOMC:
A Computational Linguistics Approach
Stephen Hansen, Michael McMahon, Andrea Prat
QJE, 2018

Look at deliberations of the Federal Open Market Committee.

➔ Key decisions on interest rate and money supply through control of the Fed's buying and selling of United States Treasury securities.

- Deliberations were recorded and transcribed just to aid with drafting minutes (since 1970's).
- But in 1993 Fed was forced to open all past and future deliberations (with 5 year delay for future).

Transparency and Deliberation Within the FOMC:
A Computational Linguistics Approach
Stephen Hansen, Michael McMahon, Andrea Prat
QJE, 2018

DiD around the policy change to see if the new transparency led members:

- Acquire and share more information on the economy. Interpreted under “career concerns” theory as them working harder due to knowledge that history will judge them. (“discipline”)
- Conform more to the prevailing opinion within the room so as to not be at risk of being singled out if wrong.

Transparency and Deliberation Within the FOMC:
A Computational Linguistics Approach
Stephen Hansen, Michael McMahon, Andrea Prat
QJE, 2018

Specifically, they used **topic modeling (Latent Dirichlet Allocation)** with the DiD to see the extent to which transparency increased certain topics and issues.

What they see:

- More economic topic coverage and references to data in early stage deliberations. (they place in line with “discipline” interpretation).
- Less discourse, less economic topic coverage, and stated opinions being closer to Fed Chair (Greenspan) in late stage deliberations. (they place in line with “conformity” interpretation).

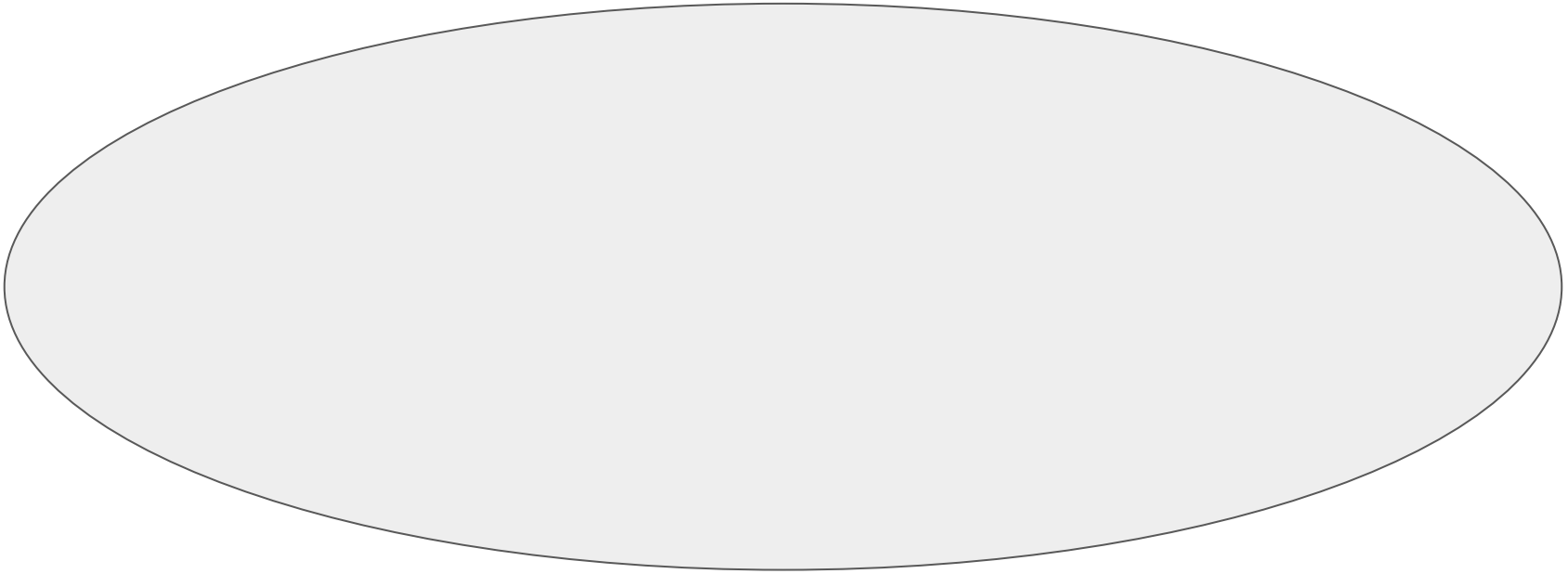
Knowledge Fit

This is part of a project looking at the relationship between researcher mobility, topic evolution, and productivity.

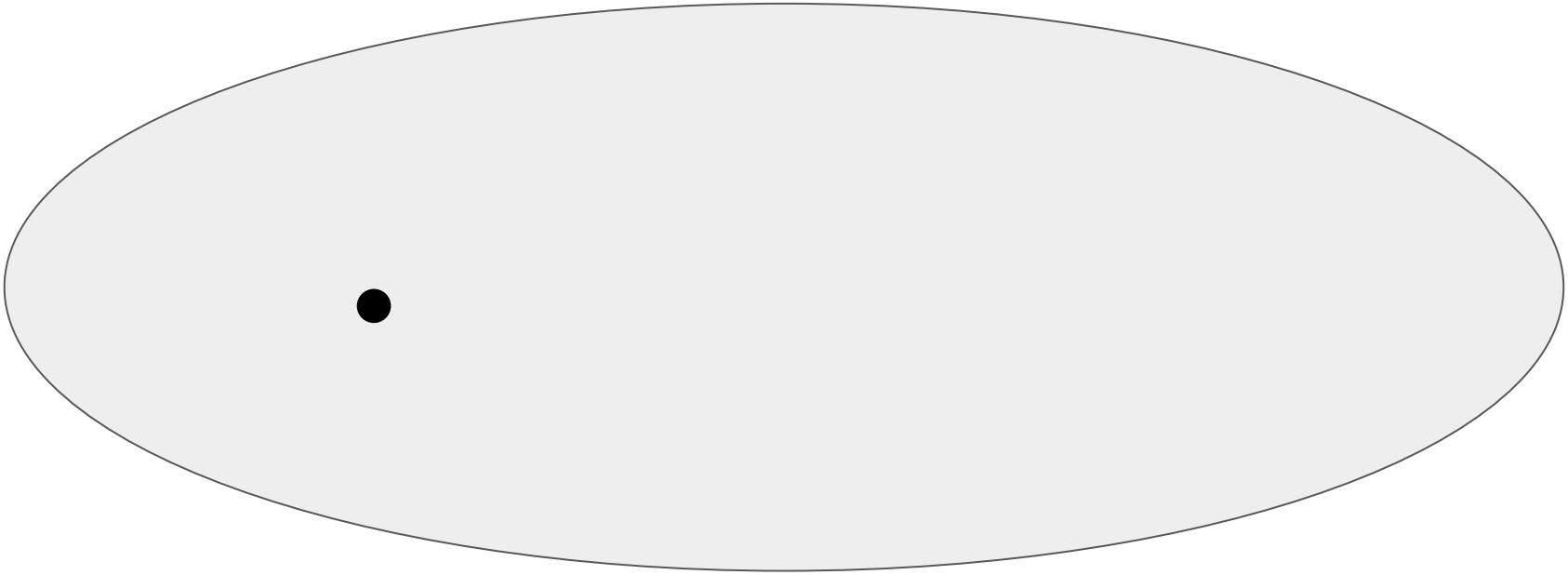
A good portion is descriptive. *i.e*. patterns of research interest evolution across career events.

But are also estimating effect of “knowledge fit” on output.

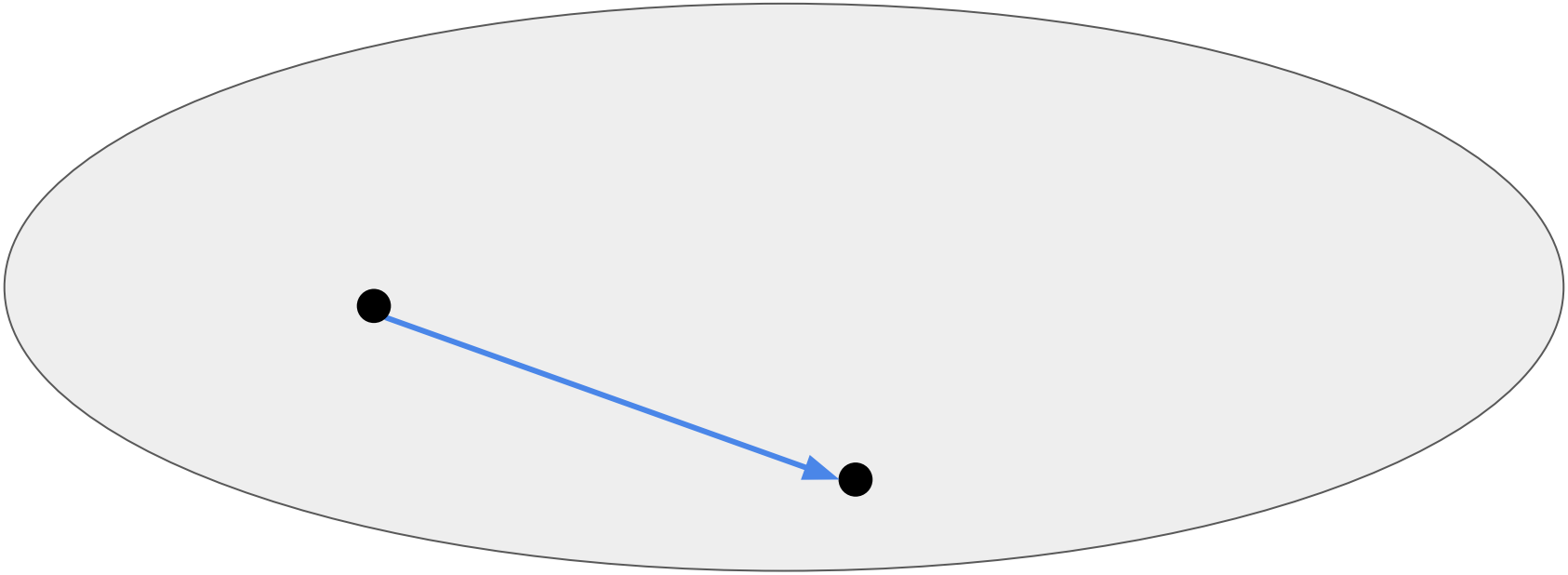
Knowledge Fit



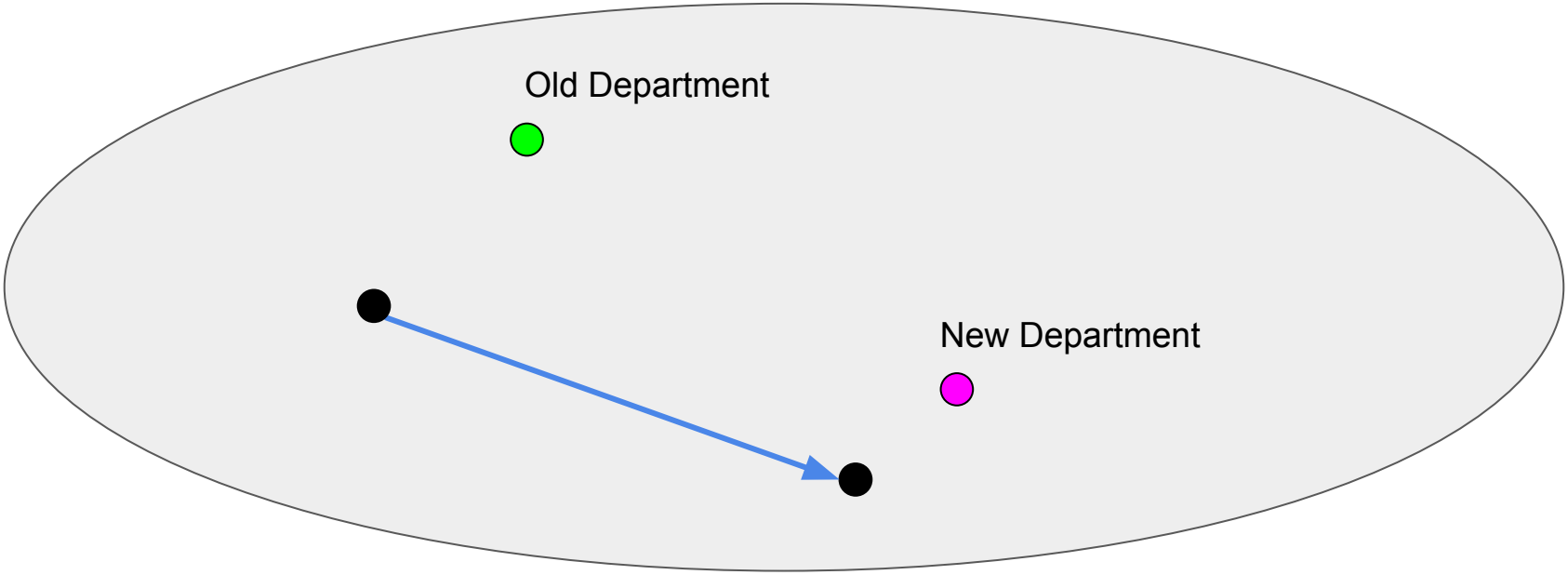
Knowledge Fit



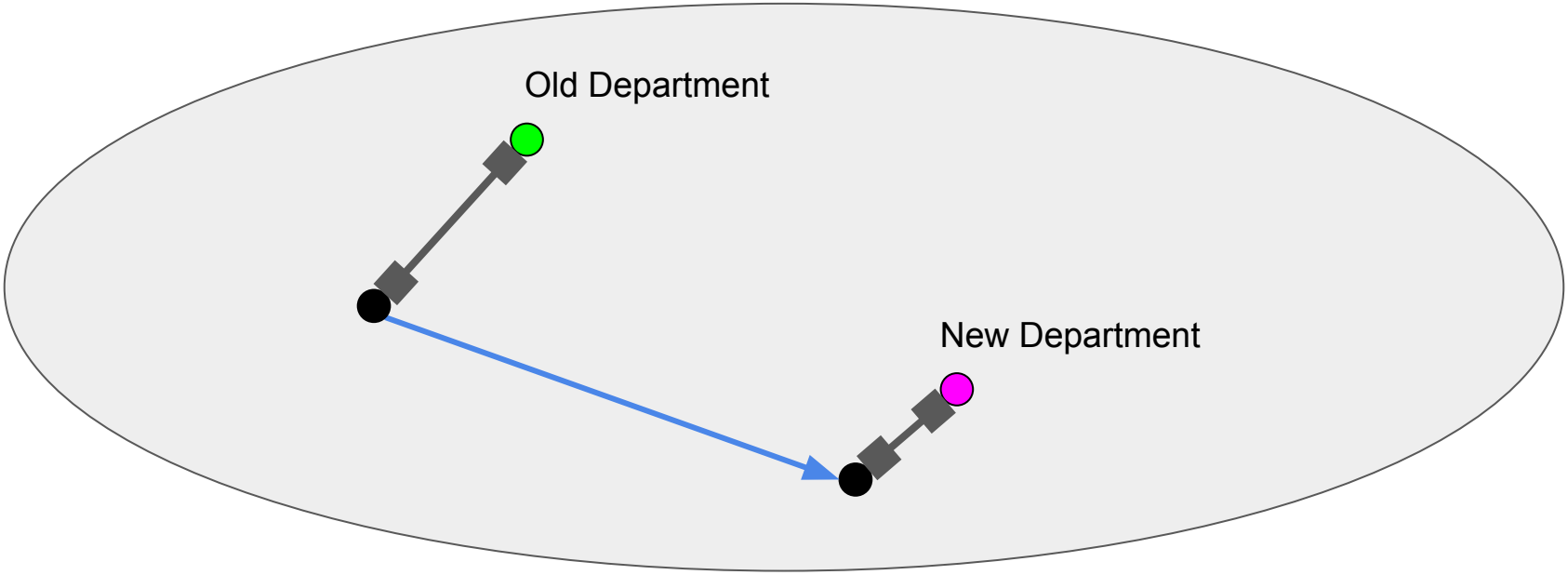
Knowledge Fit



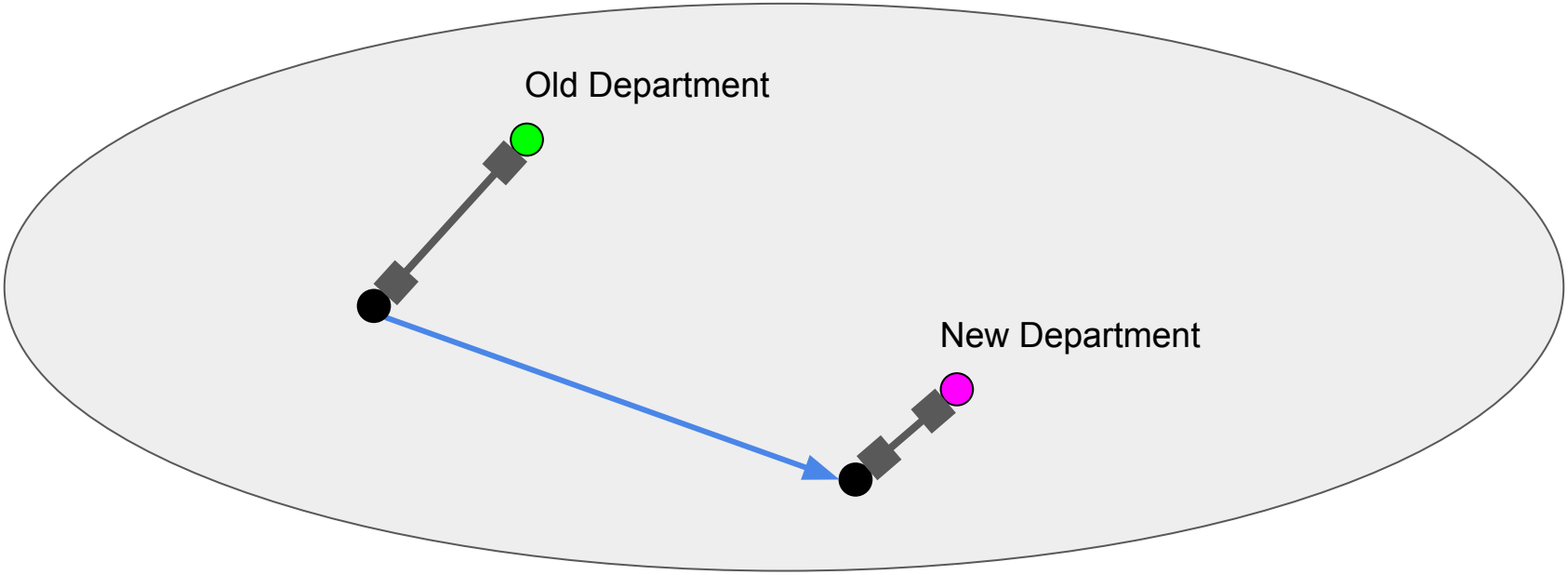
Knowledge Fit



Knowledge Fit



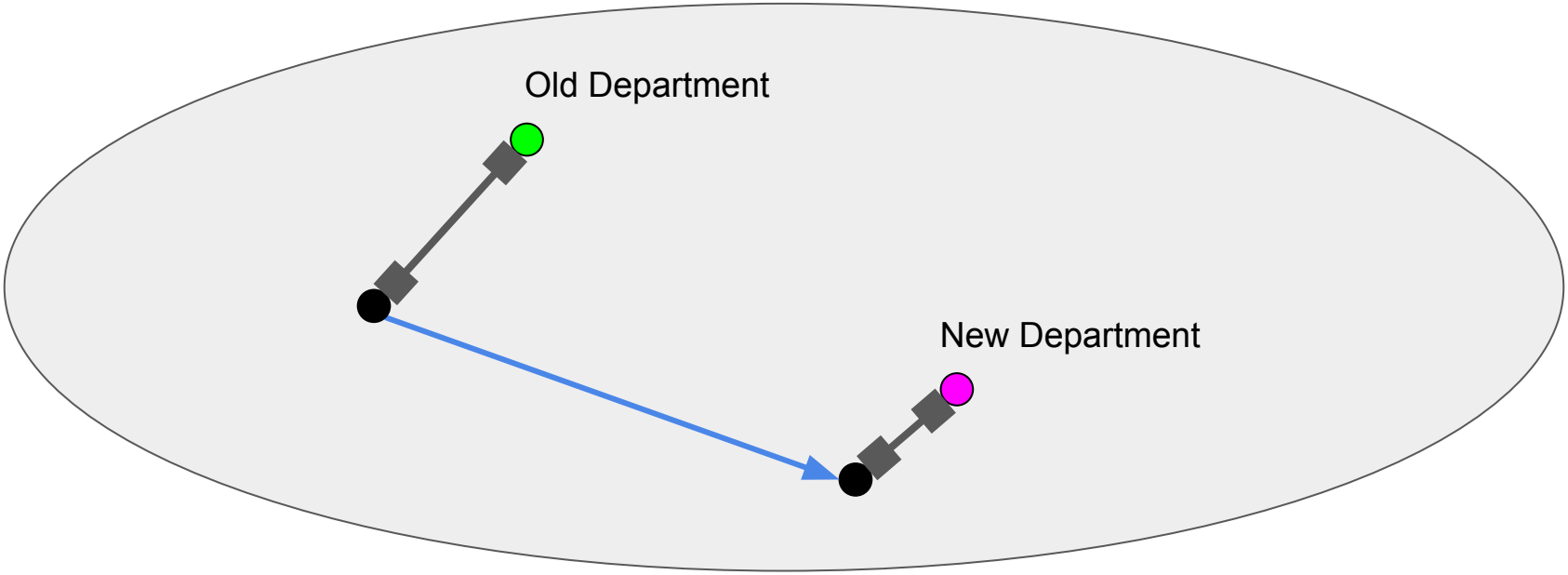
Knowledge Fit



Productivity = Researcher Specific Factors x Department Characteristics x Knowledge Fit

$$Q_{i,d} = L_i^{\theta} K_d^{\varphi} F_{i,d}^{\phi}$$

Knowledge Fit



Productivity = Researcher Specific Factors x Department Characteristics x Knowledge Fit

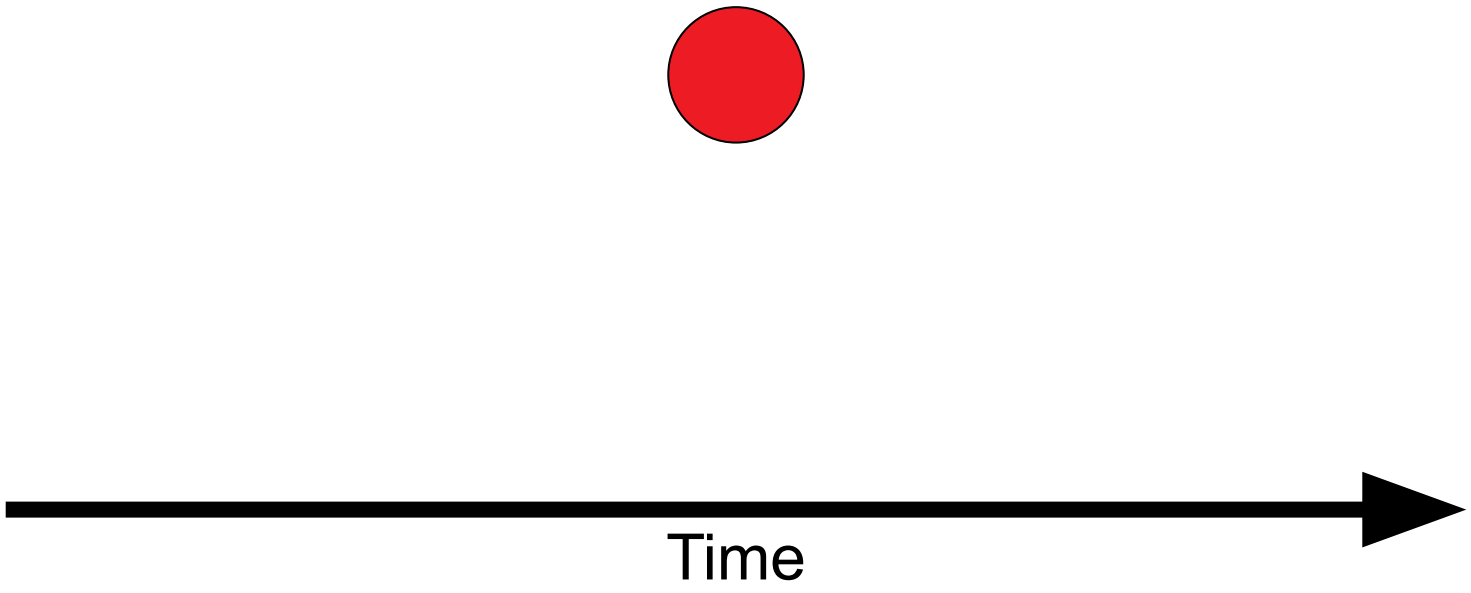
$$Q_{i,d} = L_i^{\theta} K_d^{\varphi} F_{i,d}^{\phi}$$

ΔProductivity = φ x ΔDepartment Characteristics + ϕ x ΔKnowledge Fit

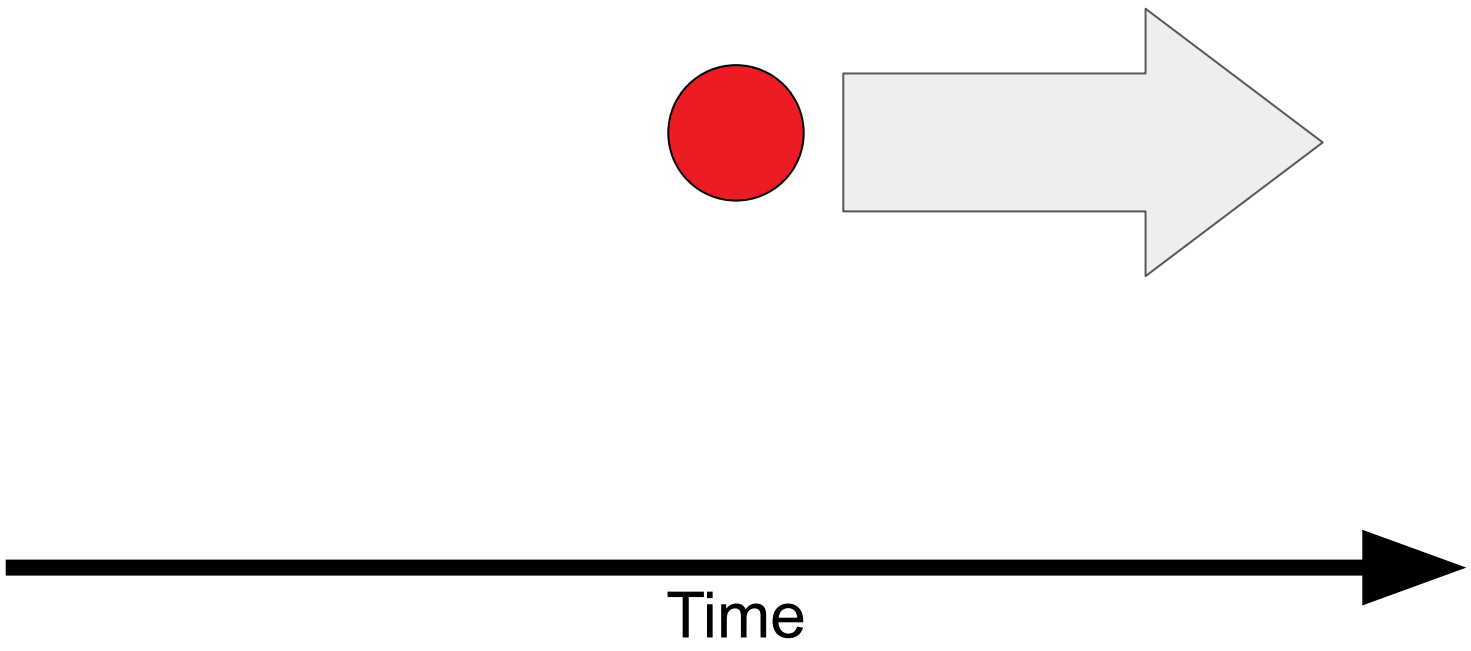
$$\Delta q_{i,d \to d'} = \varphi \Delta k_{d \to d'} + \phi \Delta f_{i,d \to d'}$$

Conceptual Influence

Conceptual Influence

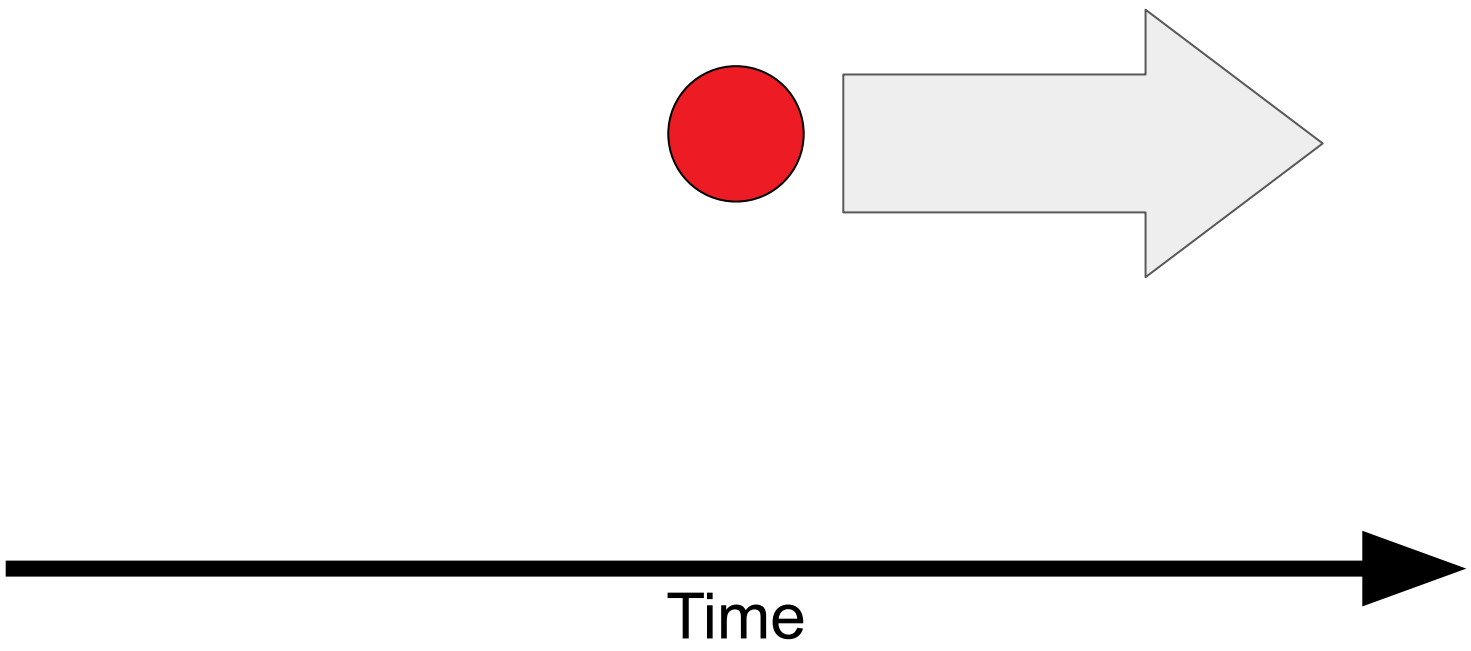


Conceptual Influence



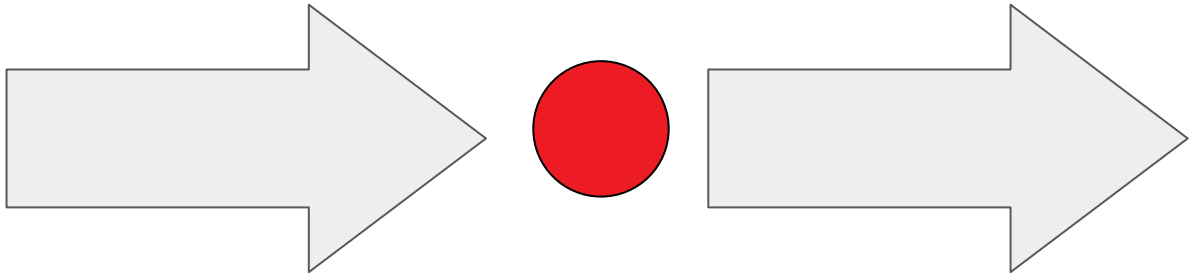
Conceptual Influence

How effectively are the ideas of a paper pushed forward?



Conceptual Influence

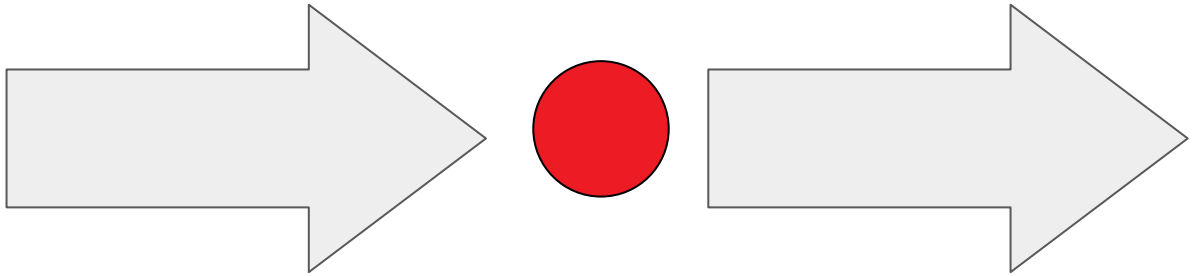
How effectively are the ideas of a paper pushed forward?



Conceptual Influence

How effectively has it integrated previous ideas?

How effectively are the ideas of a paper pushed forward?

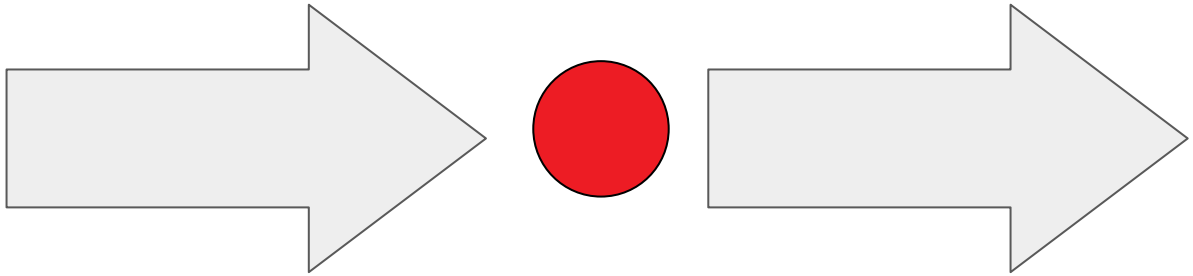


Time

Conceptual Influence

How effectively has it integrated previous ideas?

How effectively are the ideas of a paper pushed forward?

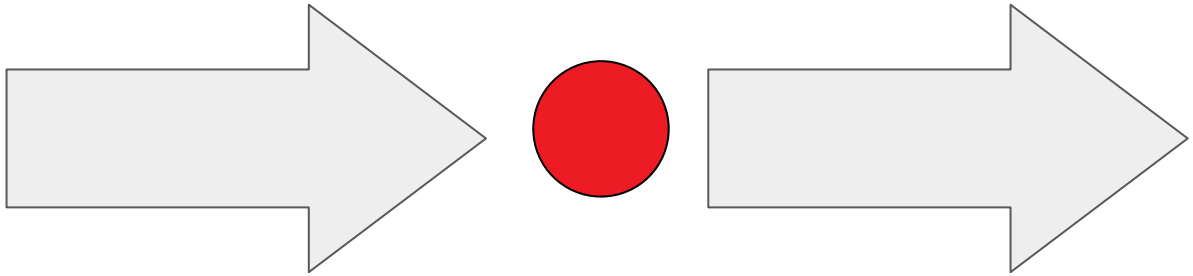


To what extent does it represent a pivot between new and old?

Conceptual Influence

How effectively has it integrated previous ideas?

How effectively are the ideas of a paper pushed forward?

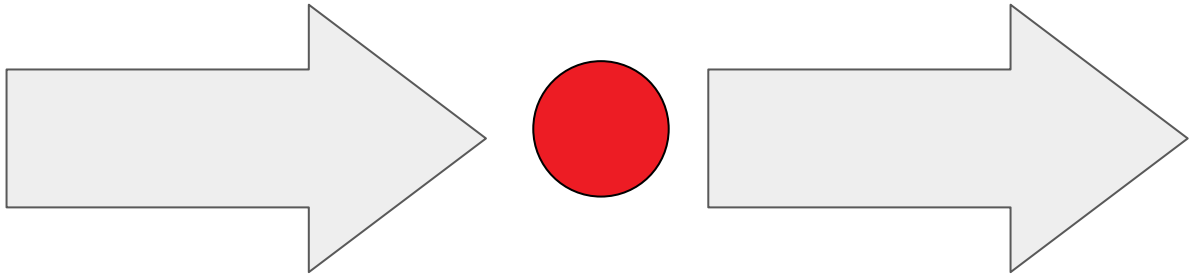


To what extent does it represent a pivot between new and old?

Conceptual Influence

How effectively has it integrated previous ideas?

How effectively are the ideas of a paper pushed forward?



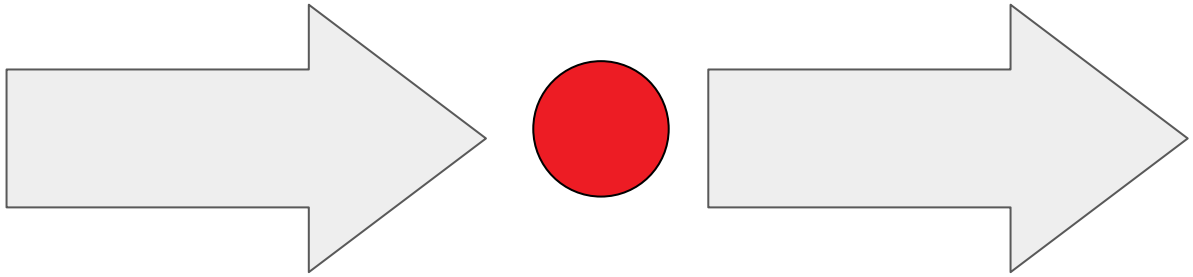
To what extent does it represent a pivot between new and old?

How big of a gap does it bridge?

Conceptual Influence

How effectively has it integrated previous ideas?

How effectively are the ideas of a paper pushed forward?



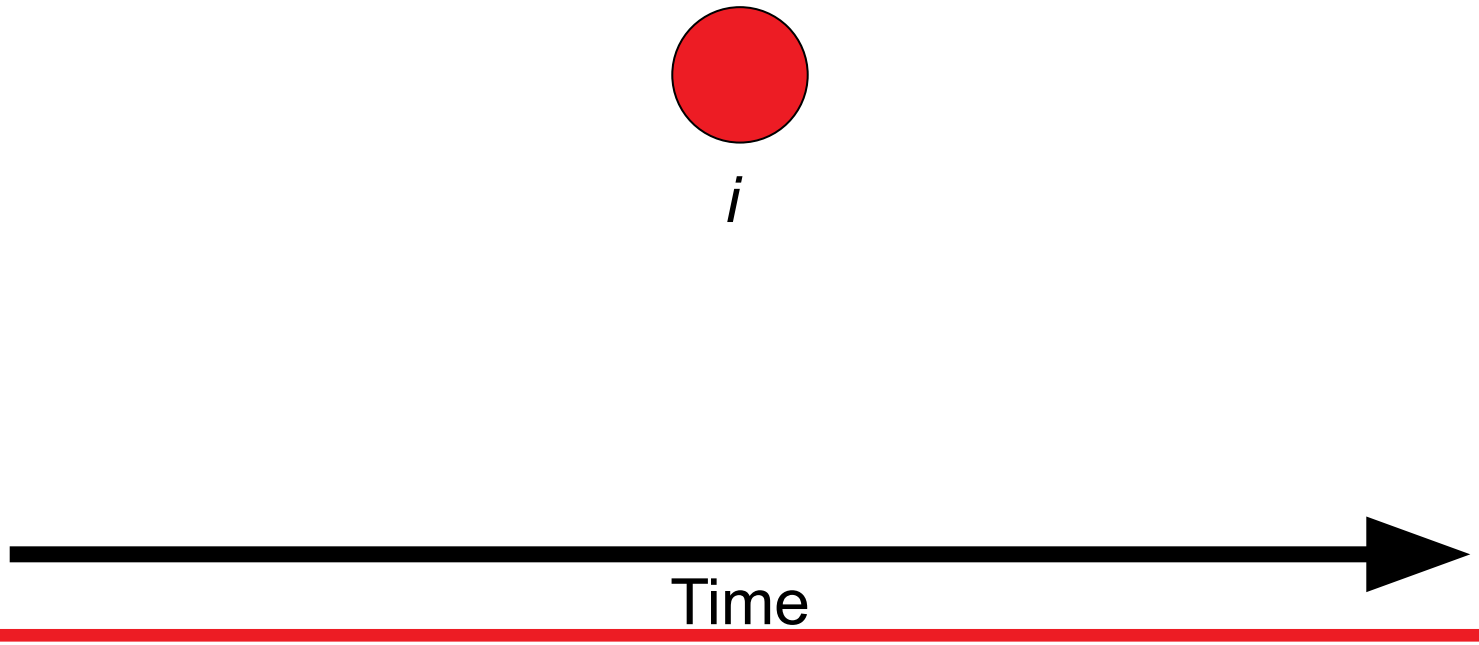
To what extent does it represent a pivot between new and old?

How big of a gap does it bridge?

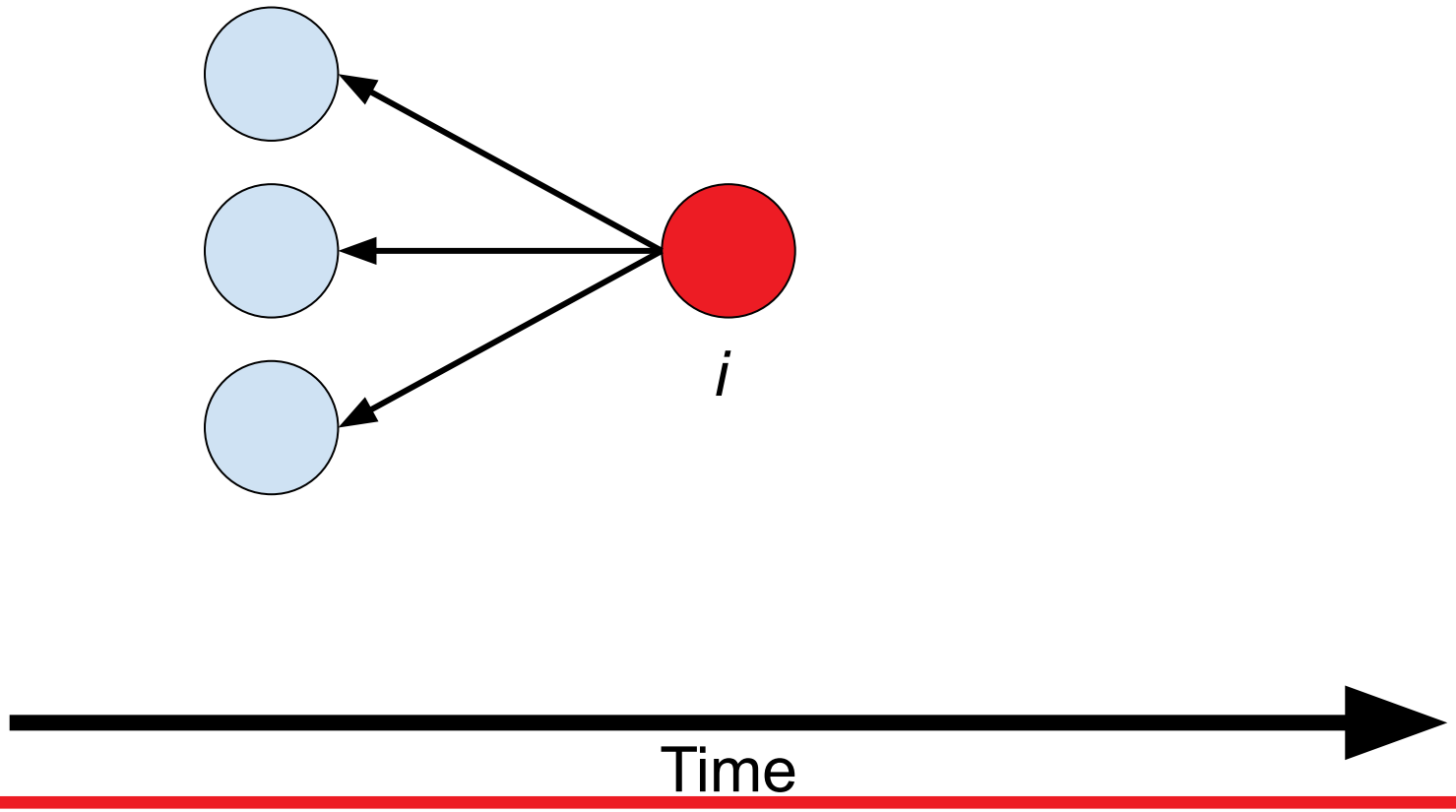
Was the pivot common? Or rare?

Time

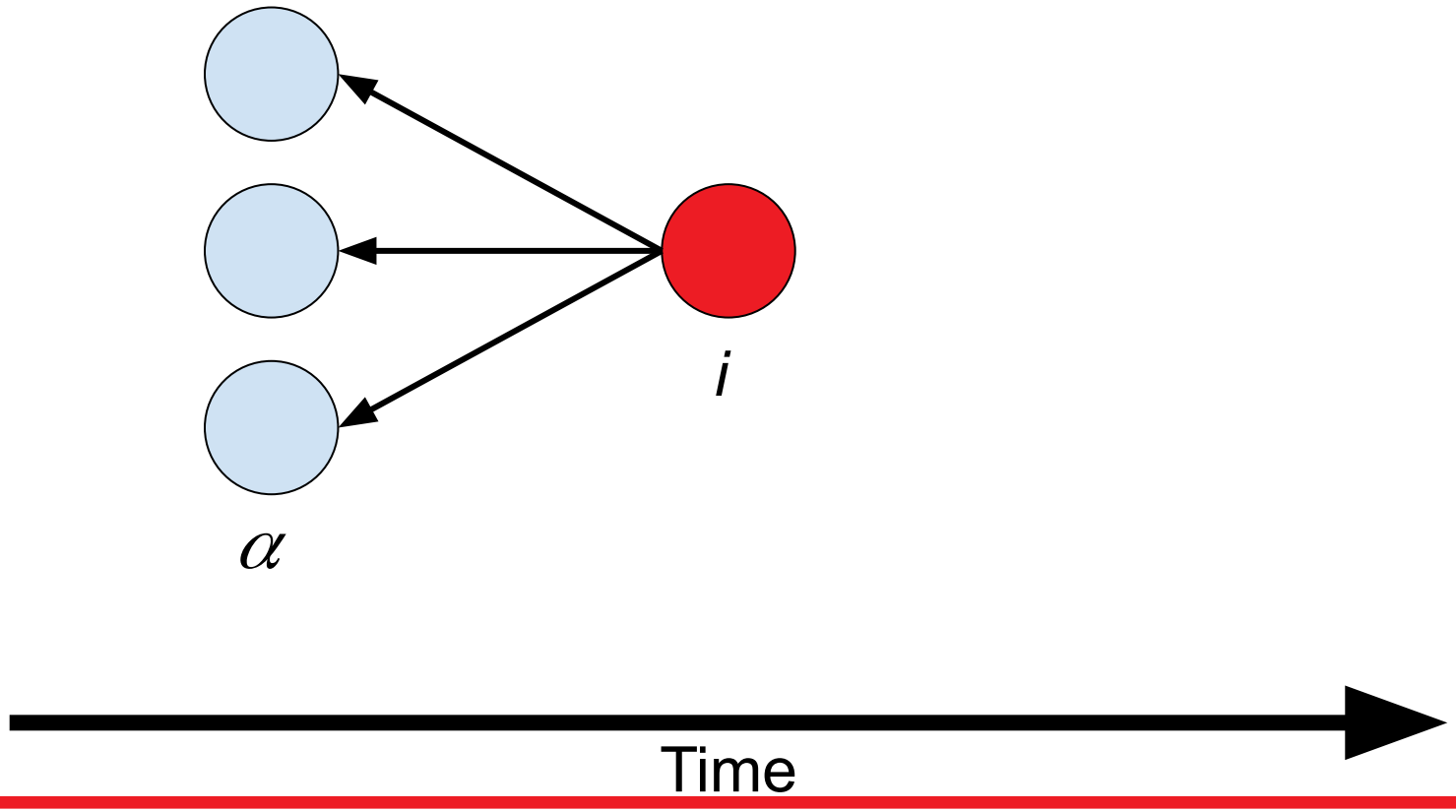
Conceptual Influence



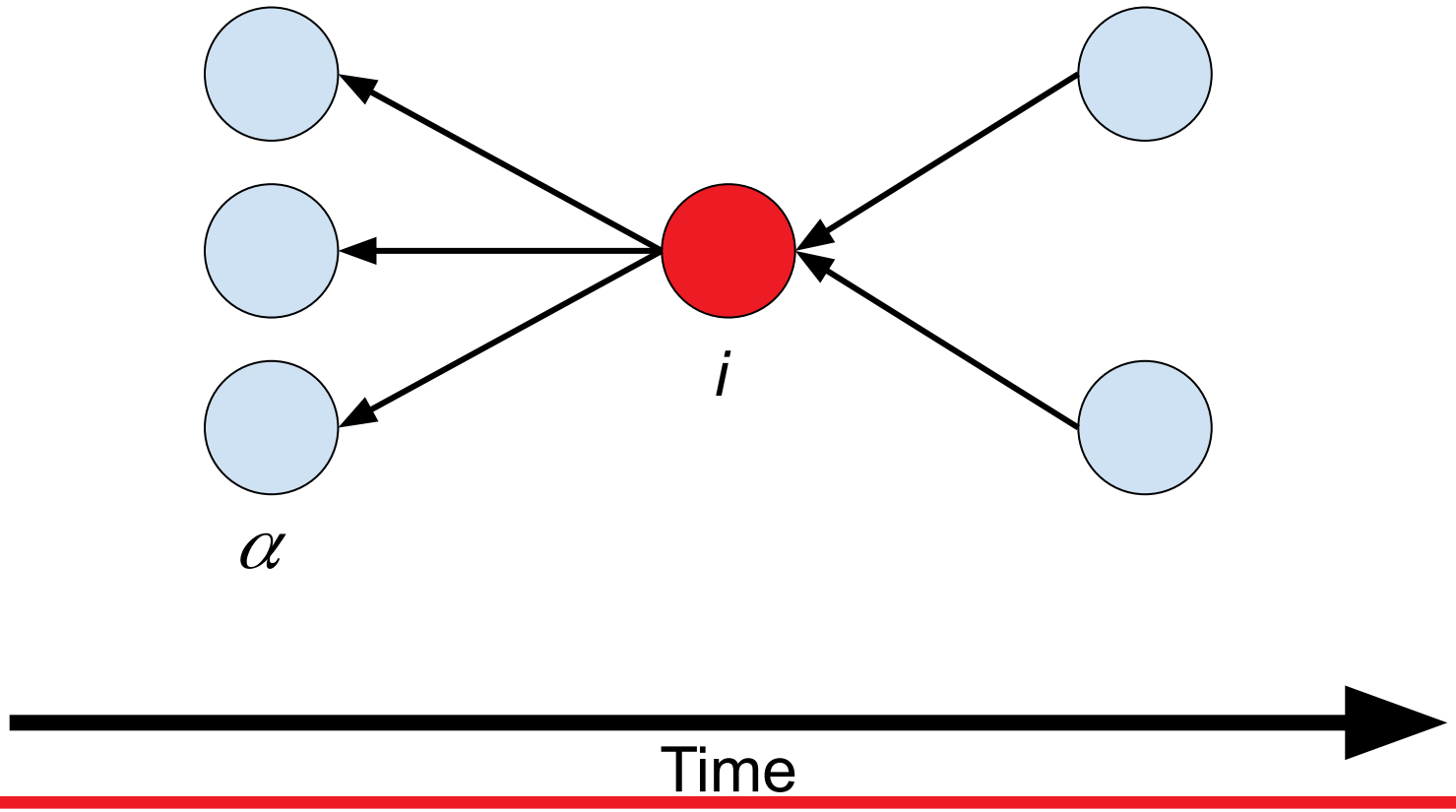
Conceptual Influence



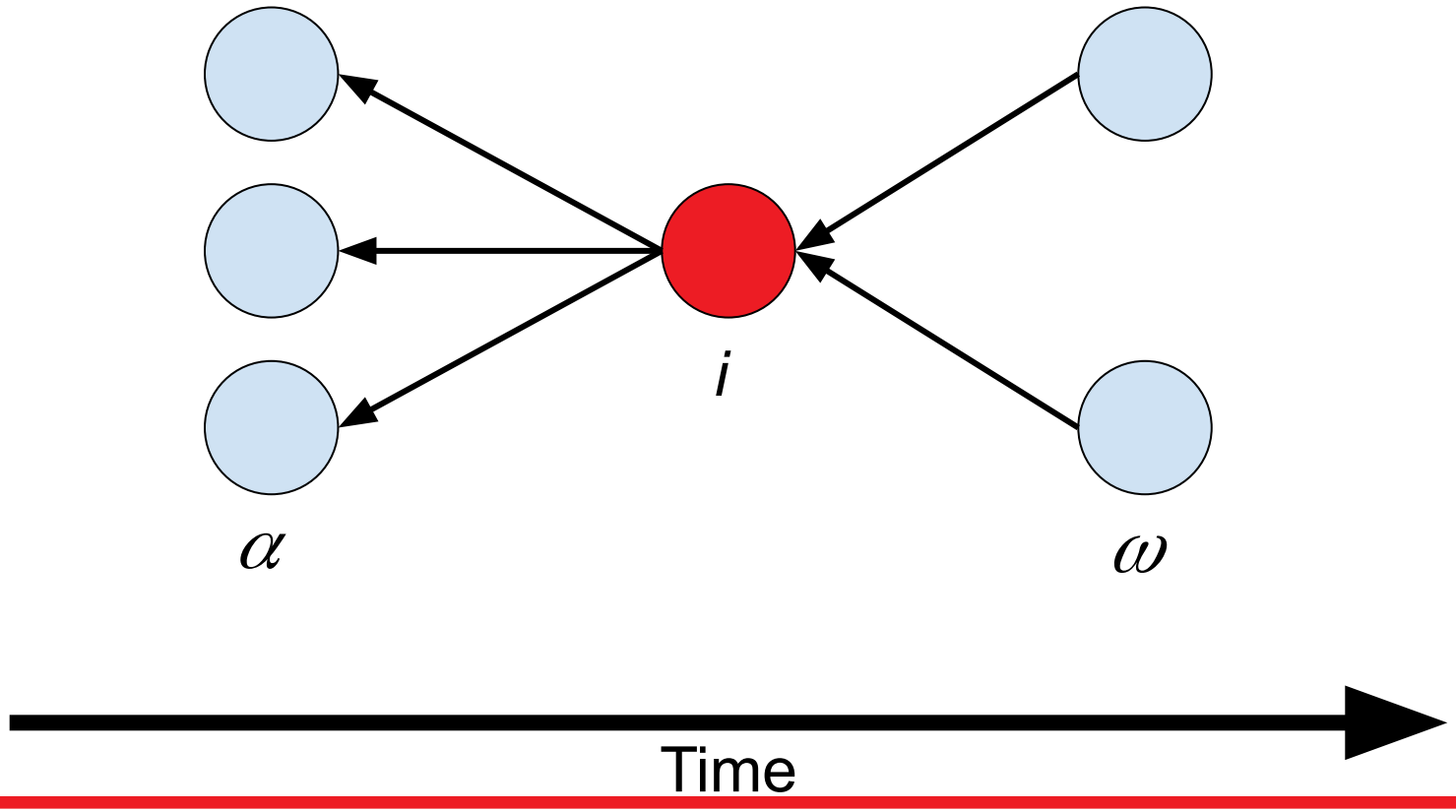
Conceptual Influence



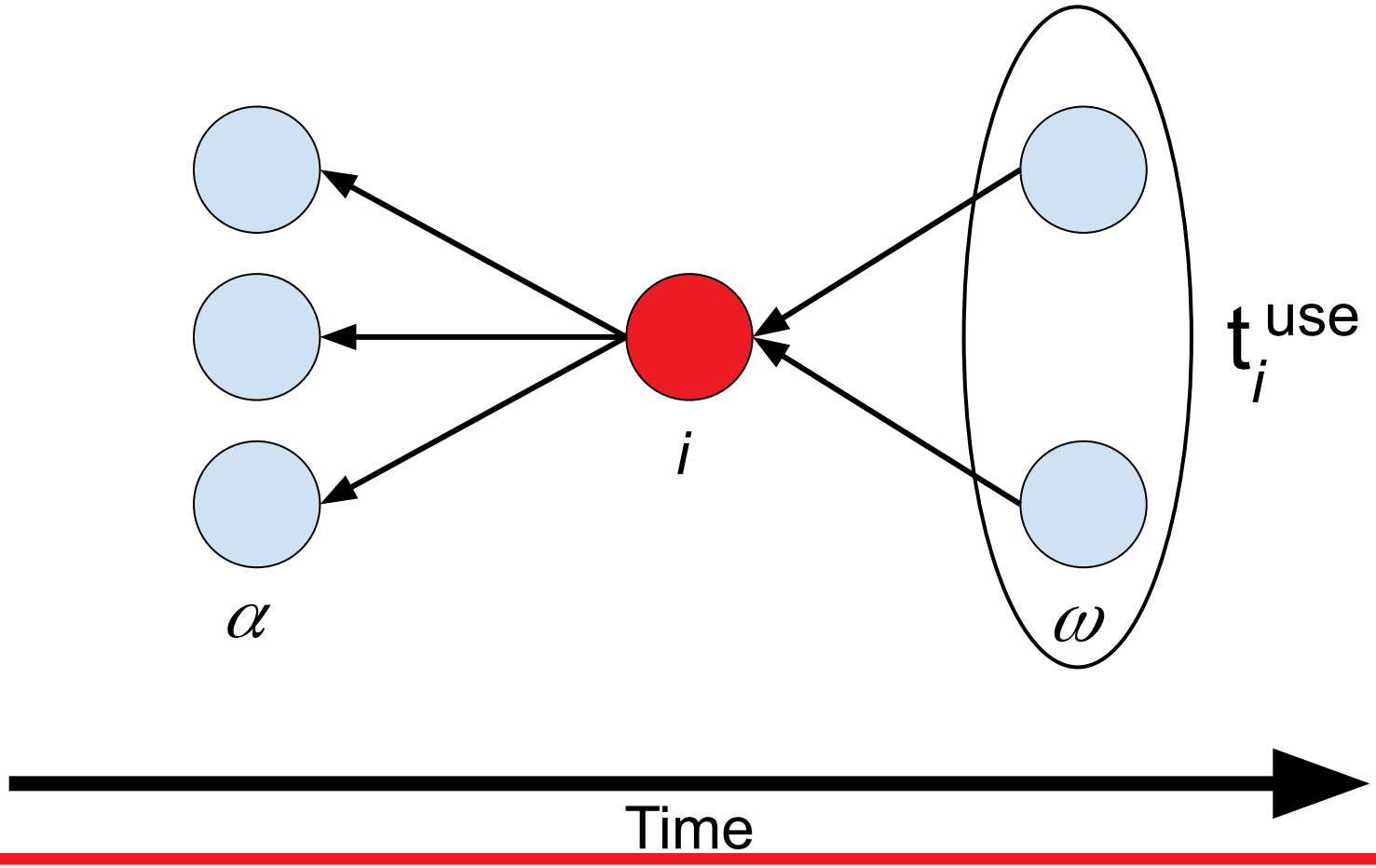
Conceptual Influence



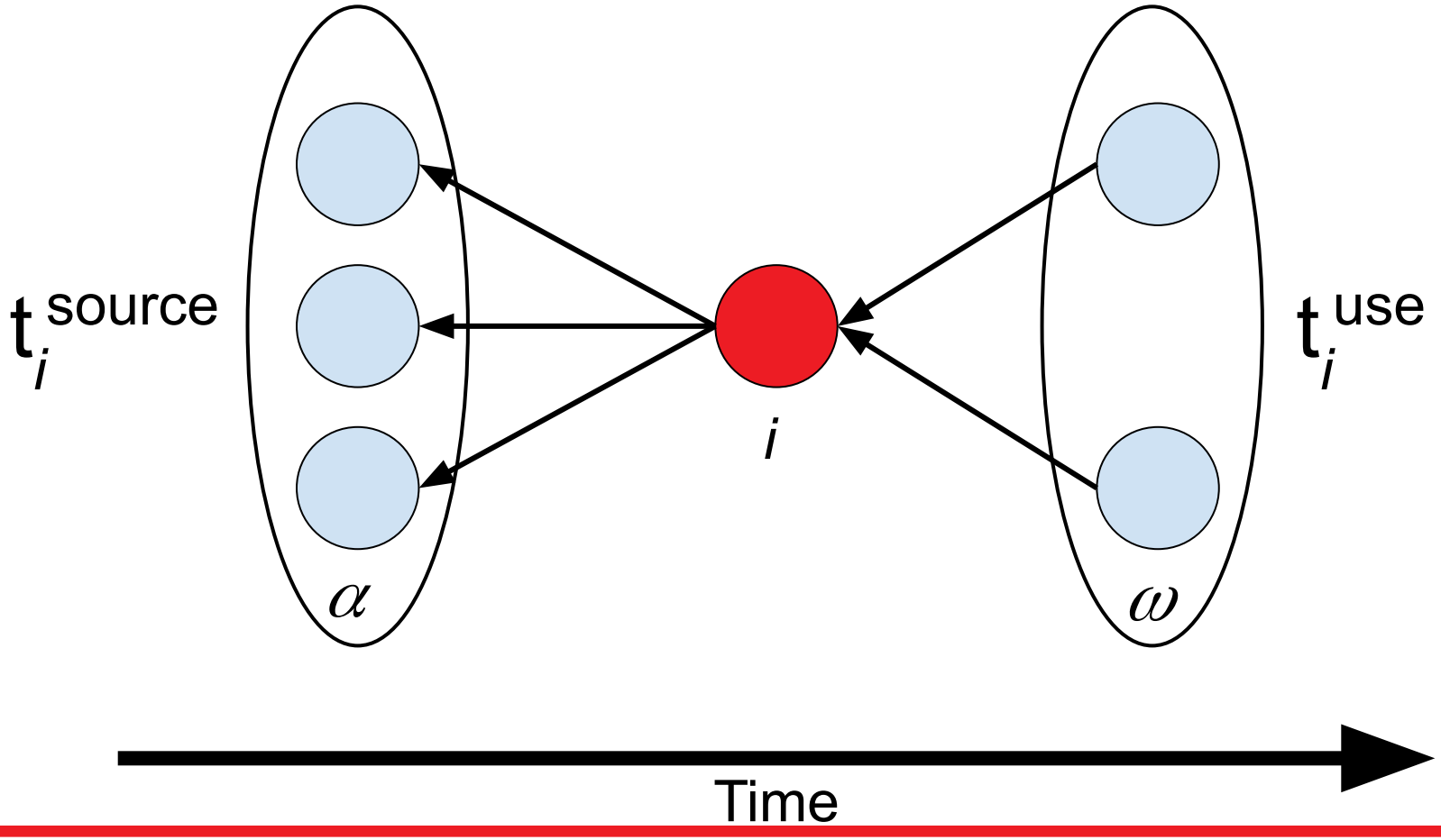
Conceptual Influence



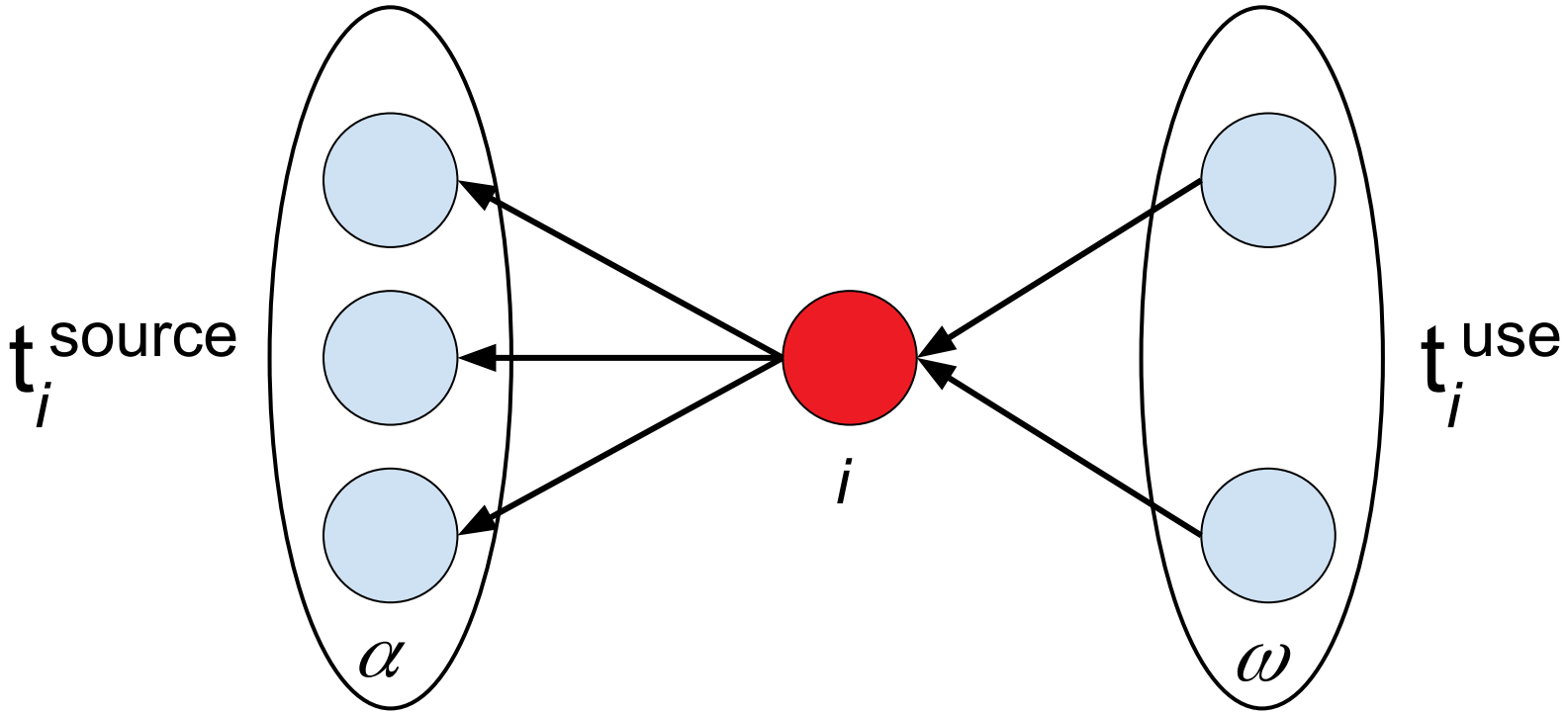
Conceptual Influence



Conceptual Influence

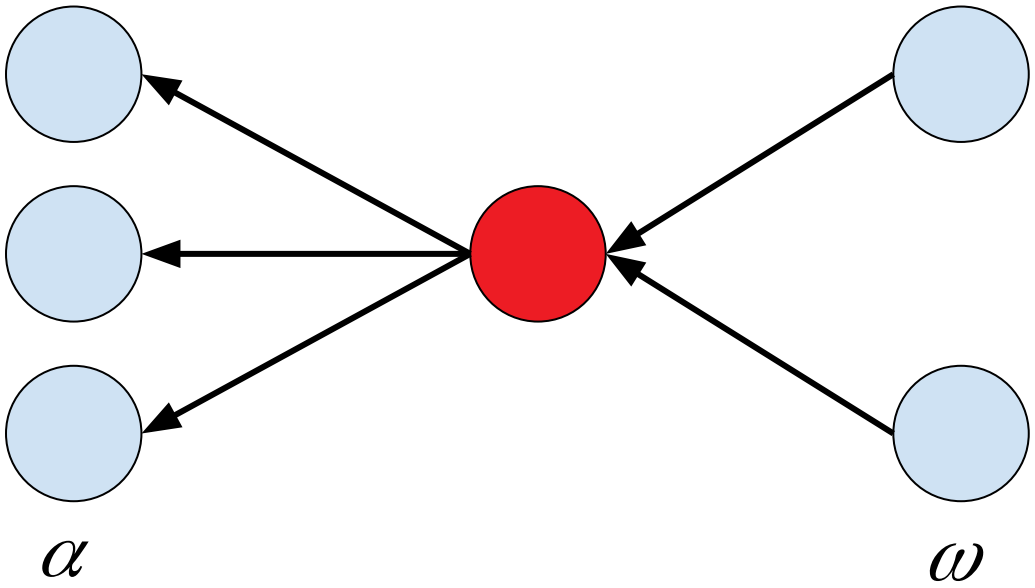


Conceptual Influence

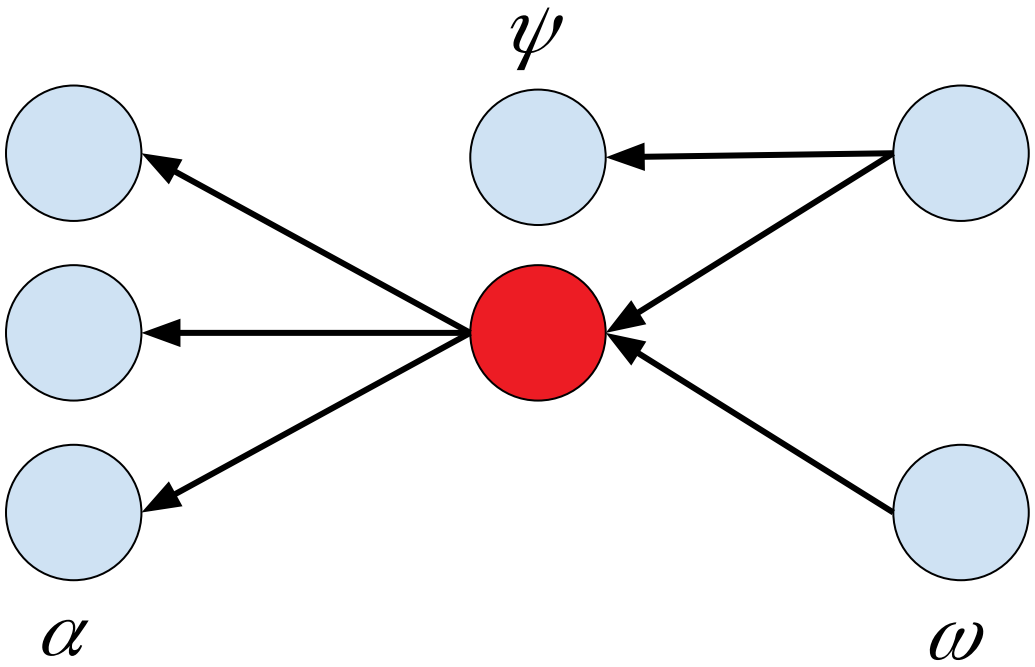


$$t_i^{source} - t_i^{use}$$

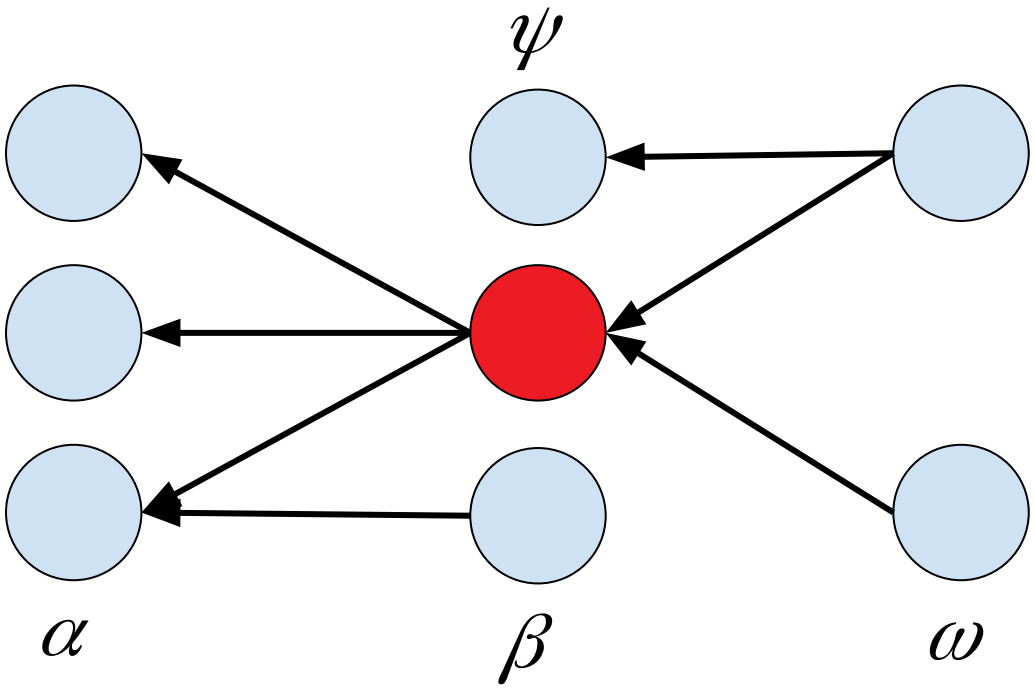
Conceptual Influence



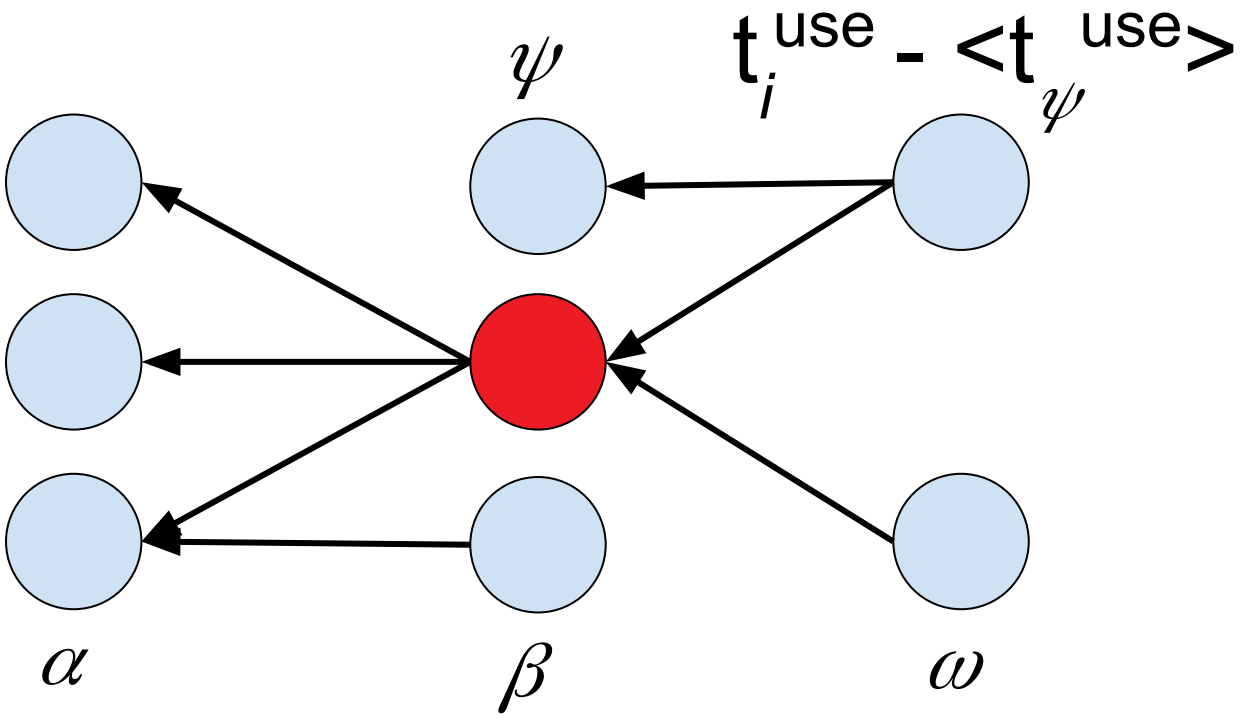
Conceptual Influence



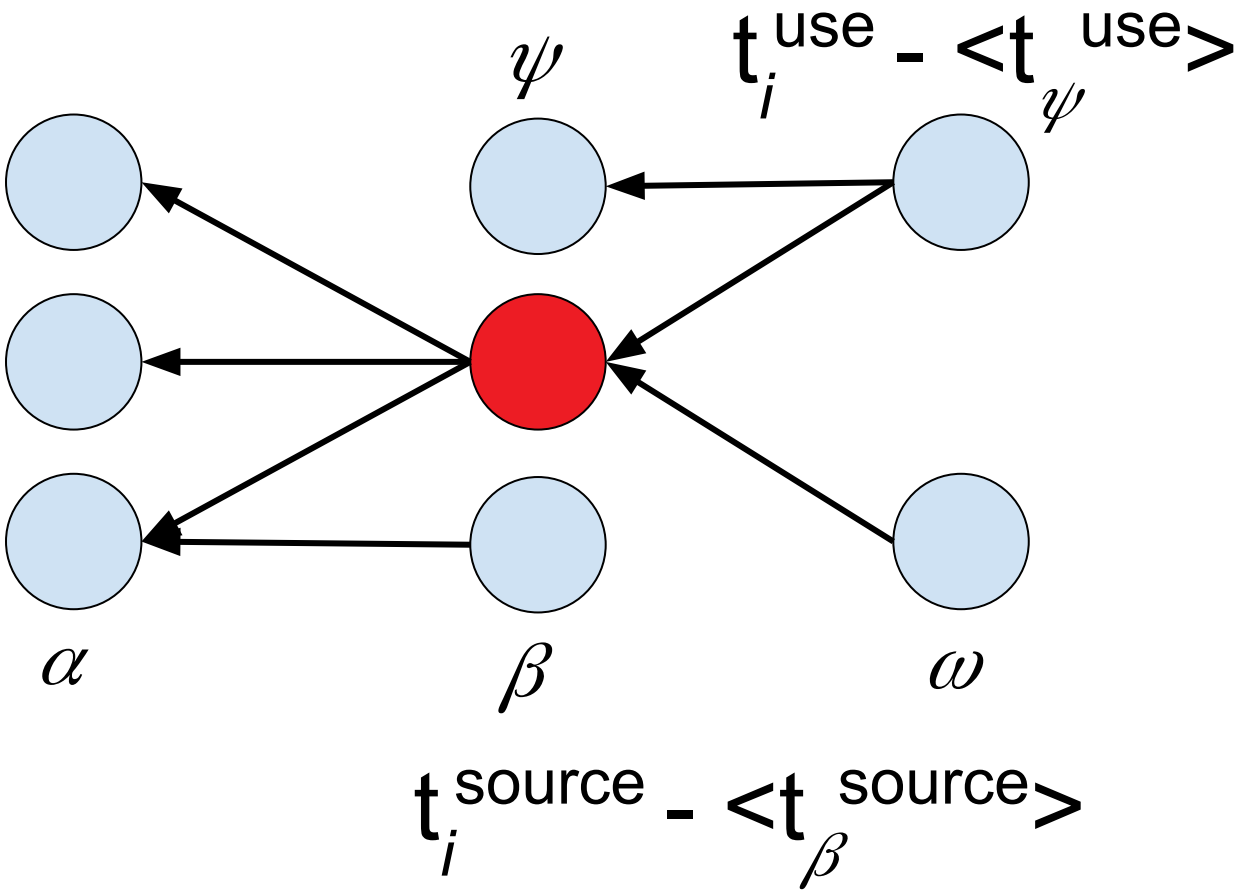
Conceptual Influence



Conceptual Influence



Conceptual Influence



Roadmap

Introduction
- Some facts and history
- What can you do with them?
- How it works – Conceptually and in practice

Two common topic models
- Non-Negative Matrix Factorization (NMF)
- word2vec & doc2vec

Using Topic Models
- Common packages
- Evaluating Robustness

How it works – Conceptually

Almost all topic modeling approaches are, at their core, trying to find **hidden semantic structures**.

Semantic structure is a elaborate way of saying that the organization and context of words (in natural language) carries meaning.

Without semantic structure:

A data table == A physical table

And:

“Man bites dog” == “Dog bites man”

Notice though that a bag-of-words based approach can only detect the first!

And also note that when talking about hidden, we generally mean to the machine. But, higher order structures may also be hidden to humans.

How it works – In Practice

Most topic modeling approaches exploit correlations to find the hidden structure.

Usually correlations between words/terms/tokens.

But **be careful** with your intuition here:

- I **do not** mean just pairwise correlations – three-point, four-point, and even higher can be exploited.
- The algorithm **is not** going to just go in and look for correlations – most are going to apply some non-linear process that exploits the underlying correlations.

We'll have a look at what this really means, but first a small aside.

Aside: Latent Space

First, consider the space in which these hidden structures are expressed.

Formally, this is the *latent space*.

With topic models this is often going to be referred to as the **topic space**.

It is just a simple *p*-dimensional Euclidean space.

The size of this space, *p*, is a free parameter. You choose it!

But isn't necessarily a great thing → no straightforward process for choosing it.

Each document gets mapped onto vector in this space. The document's "topic vector".

0.65 - Science
0.15 - University
0.30 - Publication
0.0 - Innovation
0.0 - Patent
0.0 - Firm

Now back to correlations.

How it works – In Practice

Let’s start by looking at a coupe sets of terms you may see you may see in the context of journal articles that analyze patents and publications, respectively.

How it works – In Practice

Let’s start by looking at a coupe sets of terms you may see you may see in the context of journal articles that analyze patents and publications, respectively.

inventor	citation
claims	journal
grant	author
publication	abstract
application	publication
priority	novelty
patent	bibliometric
citation	impact
non-patent literature	research

How it works – In Practice

Let's start by looking at a coupe sets of terms you may see you may see in the context of journal articles that analyze patents and publications, respectively.

inventor	citation
claims	journal
grant	author
publication	abstract
application	publication
priority	novelty
patent	bibliometric
citation	impact
non-patent literature	research

So it is probably fairly straight forward to see how a model would use the simple co-occurrence data. Essentially just clustering using counts.

How it works – In Practice

Let's start by looking at a coupe sets of terms you may see you may see in the context of journal articles that analyze patents and publications, respectively.

inventor
claims
grant
publication
application
priority
patent
citation
non-patent literature

citation
journal
author
abstract
publication
novelty
bibliometric
impact
research

So it is probably fairly straight forward to see how a model would use the simple co-occurrence data. Essentially just clustering using counts.

But there are a few in both categories. Using *ex*. cosine similarity, those would contribute to similarity.

How it works – In Practice

Let's start by looking at a coupe sets of terms you may see you may see in the context of journal articles that analyze patents and publications, respectively.

inventor
claims
grant
publication
application
priority
patent
citation
non-patent literature

citation
journal
author
abstract
publication
novelty
bibliometric
impact
research

So it is probably fairly straight forward to see how a model would use the simple co-occurrence data. Essentially just clustering using counts.

But there are a few in both categories. Using *ex*. cosine similarity, those would contribute to similarity.

However, topic models also leverage higher order relations:
(citation|inventor, patent) != (citation|journal)

How it works – In Practice

Let's start by looking at a coupe sets of terms you may see you may see in the context of journal articles that analyze patents and publications, respectively.

inventor
claims
grant
publication
application
priority
patent
citation
non-patent literature

citation
journal
author
abstract
publication
novelty
bibliometric
impact
research

So it is probably fairly straight forward to see how a model would use the simple co-occurrence data. Essentially just clustering using counts.

But there are a few in both categories. Using *ex*. cosine similarity, those would contribute to similarity.

However, topic models also leverage higher order relations:
(citation|inventor, patent) != (citation|journal)

(a, b| c, d, e) != (a, b| d, e, f)

Roadmap

Introduction
- Some facts and history
- What can you do with them?
- How it works – Conceptually and in practice

Two common topic models
- Non-Negative Matrix Factorization (NMF)
- word2vec & doc2vec

Using Topic Models
- Common packages
- Evaluating Robustness

Non-negative Matrix Factorization

NMF is a conceptually straightforward way of doing topic modeling.

It's roots actually lie in multivariate analysis and is used for many data other than text (signal processing, networks, gene expression, *etc*.).

The starting point is the Document-Term matrix, **V**.

V =

Term 1	Term 2	Term 3		
3	0	2	···	Doc 1
0	2	1	···	Doc 2
1	2	7	···	Doc 3
⋮	⋮	⋮	⋱	

This is a *m* x *n* matrix.

m documents. *n* terms (or features).

Can do any of the transforms you wish.

ex. tf, tf-idf, n-gramming *etc*.

In fact, the terms need not be word, or even text based at all.

NMF

NMF assumes the Document-Term matrix (**V**) is generated as the product of two other matrices **W** and **H**.

$$\textit{i.e. } \mathbf{V = WH}$$

W is a Document-Topic matrix and **H** is a Topic-Term matrix.

To be explicit:

- **W** tells us how much of each topic each document “has”.
  - It has dimension m x p. (recall p is dimension of latent space).
- **H** tells us how “frequently” each term should appear in documents “belonging to” each, specific, topic.
  - It has dimension p x n.

And we make the assumption that neither have any negative entries.

So **W** and **H** contain the information we want, so we should solve for them!

NMF – getting **W** and **H**

We get W and H by minimizing:

$$\|\mathbf{V} - \mathbf{WH}\|^2$$

But this is a very complex landscape, with many local minima.

So it gets solved with many of the techniques that have been developed for such problems.

ex. Fast Gradient Descent, Gibbs sampling, Simulated annealing, *etc*.

It is important to keep in mind though that there can be (huge) variation in the **W** and **H** arising from different local minima!

So, especially for many latent dimensions, you really have to think carefully about whether or not you trust the results.

NMF – Wrap up

NMF is a fairly straightforward and interpretable topic model.

It basically assumes the following generative process:

1. Each document has an intrinsic topic profile (vector).
2. Within a document, each word appears with “frequency” proportional to how strongly associated that word is to each topic, multiplied by how much of that topic the document “has”.

Pros:

- Interpretation is fairly straightforward
- It is *flexible* in that it allows things other than words to be the features

Cons:

- The forced linearity can limit the range of relations it can exploit
- Questionable robustness. Many local minima. (more later)

Roadmap

Introduction
- Some facts and history
- What can you do with them?
- How it works – Conceptually and in practice

Two common topic models
- Non-Negative Matrix Factorization (NMF)
- word2vec & doc2vec

Using Topic Models
- Common packages
- Evaluating Robustness

word2vec & doc2vec

word2vec and doc2vec formulate topic modeling as a supervised ML problem.

But how!?!?! We have no ground truth for the topics!

Beauty is, word2vec isn't trying to predict anything related to the latent space.

word2vec & doc2vec

word2vec formulates a problem of predicting either:

A. The context of a word as it sits in a string of text.
   i.e. given a sequence: [___ ___ house ___ ___] (predict the 4 missing words)
B. Given the context, predict the missing word.
   i.e. in the sequence: [the blue ___ is big] (predict the one missing word)

Using a construction that exploits a latent space.

By way of a neural network.

And after training, the **term-topic** vectors are extracted from the NN.

Be very careful though, because this is gives us a word →topic mapping.

Not the document →topic mapping we're looking for.

word2vec & doc2vec

We could just add up the term-topic vectors for all the words in a document to create a document-topic vector.

However, this produces awful results! Mostly because it doesn’t use the fact that the same word can mean very different things in different types of documents.

word2vec & doc2vec

doc2vec is a modification of word2vec that does use that information by modifying the prediction problem to:

A. Given [___ ___ house ___ ___, Document #1], predict the missing.
B. Given [the blue ___ is big, Document #1], predict the missing.

There is nothing magical here, the document tag is just another piece of the information the NN is being given to predict the missing words.

So similar to the words in word2vec, a document-topic vector is going to be inferred for each document!

word2vec & doc2vec - Wrap up

word2vec is a NN based approach that formulates a missing word prediction problem, and then allows the NN to use a latent space to make predictions. From the parameters of the NN we can then extract word-topic vectors.

word2vec & doc2vec - Wrap up

word2vec is a NN based approach that formulates a missing word prediction problem, and then allows the NN to use a latent space to make predictions. From the parameters of the NN we can then extract word-topic vectors.

Doc2vec is a modification that allows the NN to also use a document tag to improve its prediction. And from the parameters of the NN we can also extract document-topic vectors!

word2vec & doc2vec - Wrap up

word2vec is a NN based approach that formulates a missing word prediction problem, and then allows the NN to use a latent space to make predictions. From the parameters of the NN we can then extract word-topic vectors.

Doc2vec is a modification that allows the NN to also use a document tag to improve its prediction. And from the parameters of the NN we can also extract document-topic vectors!

Pros:

- This approach works! Actually up to fairly large latent spaces.
- And is actually robust!
- And uses local context, which can be positive.

word2vec & doc2vec - Wrap up

word2vec is a NN based approach:

- That formulates a missing word prediction problem,
- Allows a NN to use a latent space to make those predictions
- And from the parameters of the NN we can then extract word-topic vectors.

doc2vec is modification of word2vec that:

- Adds a document tag to the word sequence.
- Creating a coupling between a document and each latent space dimension that can be interpreted as a topic vector.

word2vec & doc2vec - Wrap up

Pros:

- This approach works! Actually up to fairly large latent spaces.
- And is actually robust!
- And uses local context, which can be positive.

Cons:

- It works, but how. It's a Neural Network so no one really knows!
- Interpretability?
- It uses local context, so if there are some (much) longer range correlations that are important in your system, it can miss them.

Roadmap

Introduction
- Some facts and history
- What can you do with them?
- How it works – Conceptually and in practice

Two common topic models
- Non-Negative Matrix Factorization (NMF)
- word2vec & doc2vec

Using Topic Models
- Common packages
- Evaluating Robustness

Common Packages

Disclaimer: I code in Python. It is what I know best. There may be other non-Python options beyond those I mention.

Python

- Scikit Learn has NMF and LDA implementation in sklearn.decomposition
- Gensim has LDA and doc2vec (and a lot more).
  - Is a little slow because it mostly SciPy/NumPy based.
  - Has good distributed parallel implementations of many algorithms.
- As doc2vec is neural network based, it can be implemented using Theano/TensorFlow/PyTorch. Some Keras implementations around.

R has topicmodels. I've seen many people using it, but it is only LDA.

There are java libraries like MALLET and the Stanford Topic Modeling Toolbox.

Evaluating Robustness

Once you have your topic model you should be able to do two things, in theory:

1. Validate its robustness. *i.e.* make sure that when you run it multiple times with the same prior/initialization/random seed, you get more/less the same results.
2. Validate it with a "ground truth" data. *i.e.* compare your results to expectations of experts or similar.

Point 1 we have actually made some progress on.

Point 2 is more difficult (→impossible) if we work from the assumption that data was unlabeled to being with. If it were labeled, would have just used that and done a classifier!

So for point 2 all you can ever hope for is finding a clever sanity check.

Evaluating Robustness

There are some mutual information approaches for measuring how topics vary – groups of terms stay together or get broken apart – as you run/train your model multiple times.

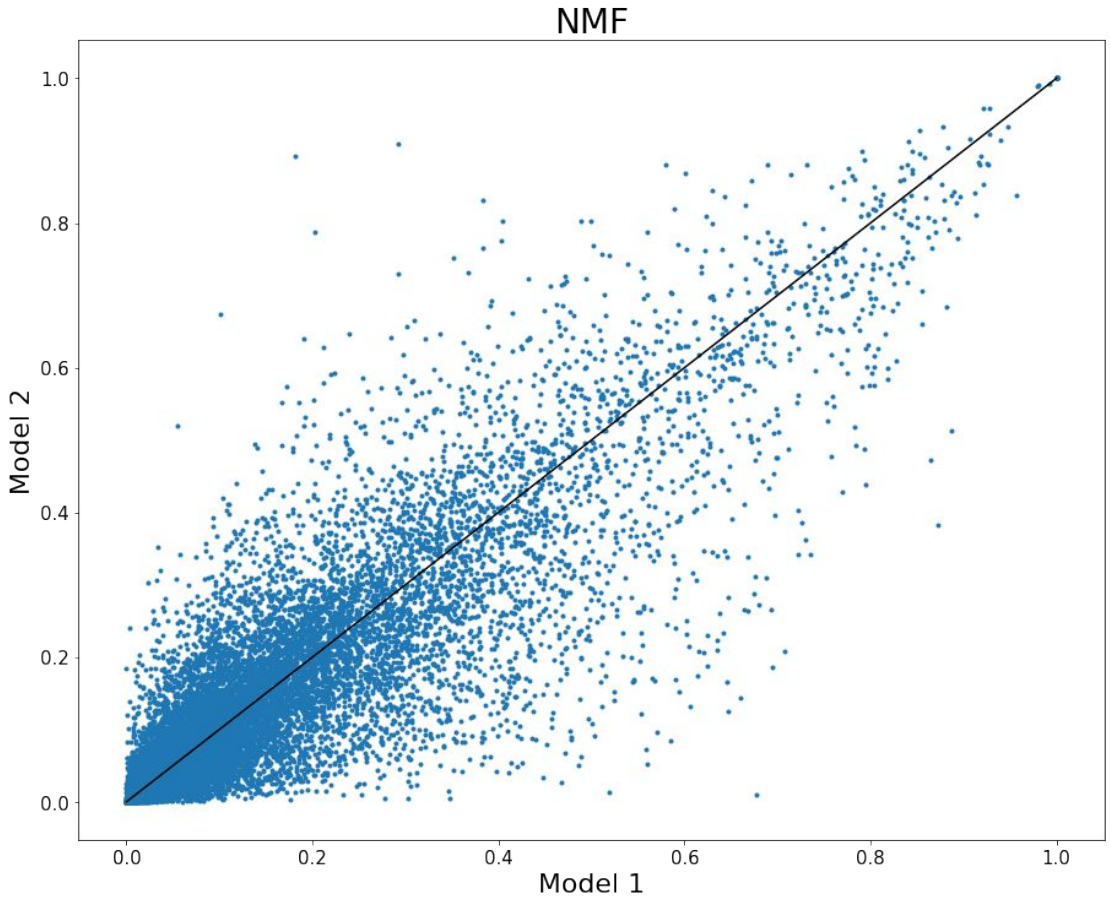
However, such approaches focus *only* on the topics themselves. Not how the topic composition of the documents change.

But the inferred relationships between the documents (*ex*. similarity) can be significantly disrupted by even relatively small shifts of terms among the topics.

After a lot of pain, frustration and approximately 8000 CPU hours we *think* we've arrived at a reasonable robustness measure based on **convergence of document-document similarity scores** (or lack thereof).

Sadly there is absolutely nothing clever here though!

Evaluating Robustness



Evaluating Robustness

Let *i* and *j* be two documents in our corpus.

Let $\mathbf{w}^t_i$, $\mathbf{w}^t_j$ be the topic vectors of *i* and *j* for a run *t*.

These vectors can be inserted into our favorite similarity measure, say cosine:

$$S^t_{i,j} = \frac{\mathbf{w}^t_i \bullet \mathbf{w}^t_j}{\|\mathbf{w}^t_i\| \, \|\mathbf{w}^t_j\|}$$

Evaluating Robustness

Let *i* and *j* be two documents in our corpus.

Let $\mathbf{w}^t_i$, $\mathbf{w}^t_j$ be the topic vectors of *i* and *j* for a run *t*.

These vectors can be inserted into our favorite similarity measure, say cosine:

$$S^t_{i,j} = \frac{\mathbf{w}^t_i \bullet \mathbf{w}^t_j}{\|\mathbf{w}^t_i\| \, \|\mathbf{w}^t_j\|}$$

Now calculate the standard deviation of each pair across the sample (N runs): $\sigma_{i,j}$

Evaluating Robustness

Let *i* and *j* be two documents in our corpus.

Let $\mathbf{w}^t_i$, $\mathbf{w}^t_j$ be the topic vectors of *i* and *j* for a run *t*.

These vectors can be inserted into our favorite similarity measure, say cosine:

$$S^t_{i,j} = \frac{\mathbf{w}^t_i \bullet \mathbf{w}^t_j}{\|\mathbf{w}^t_i\| \, \|\mathbf{w}^t_j\|}$$

Now calculate the standard deviation of each pair across the sample (N runs): $\sigma_{i,j}$

As a bulk measure, can sum $\sigma_{i,j}$ over all pairs (*i*,*j*).

Evaluating Robustness

Let *i* and *j* be two documents in our corpus.

Let $\mathbf{w}^t_i$, $\mathbf{w}^t_j$ be the topic vectors of *i* and *j* for a run *t*.

These vectors can be inserted into our favorite similarity measure, say cosine:

$$S^t_{i,j} = \frac{\mathbf{w}^t_i \bullet \mathbf{w}^t_j}{\|\mathbf{w}^t_i\| \, \|\mathbf{w}^t_j\|}$$

Now calculate the standard deviation of each pair across the sample (N runs): $\sigma_{i,j}$

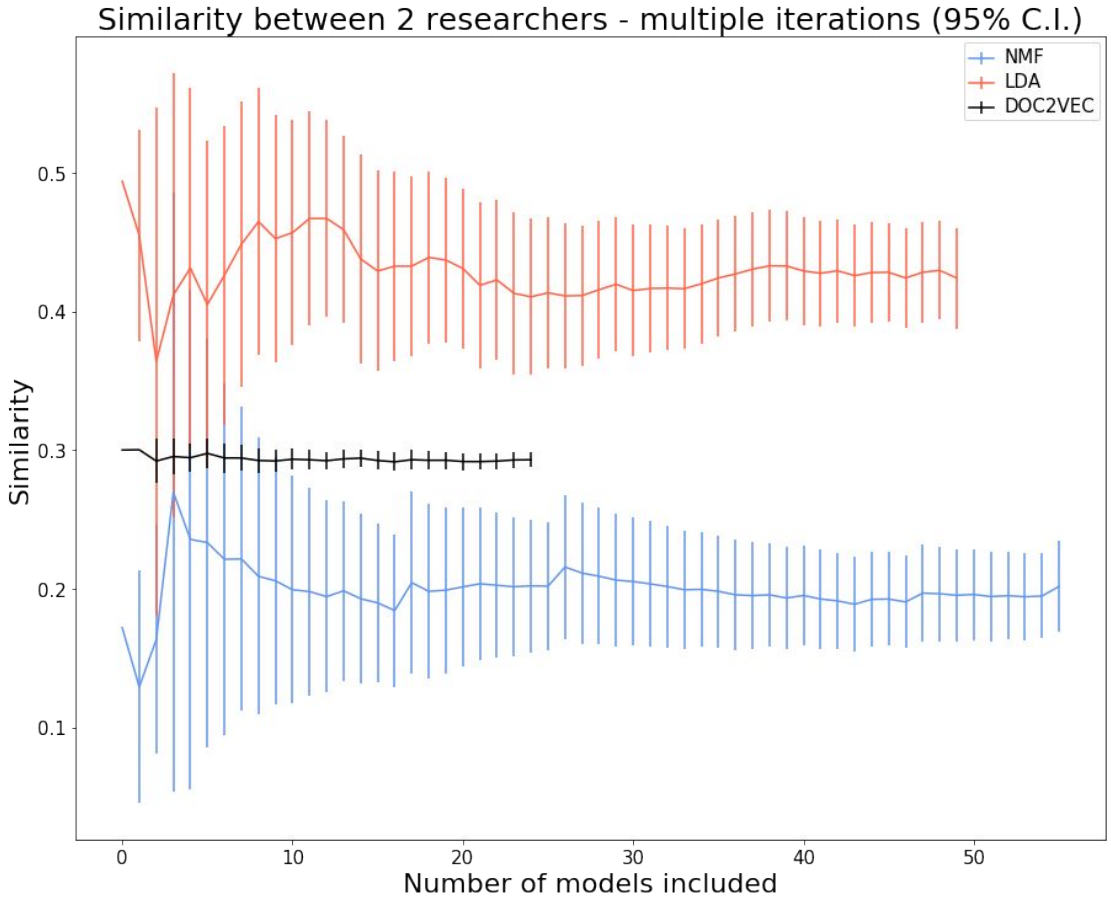
As a bulk measure, can sum $\sigma_{i,j}$ over all pairs (*i*,*j*).

And can also check the rate at which the measure decreases as function of (N).

I am not going to show this figure because it is running.

But next best thing...

Evaluating Robustness



Evaluating Robustness

But then a second problem emerges.

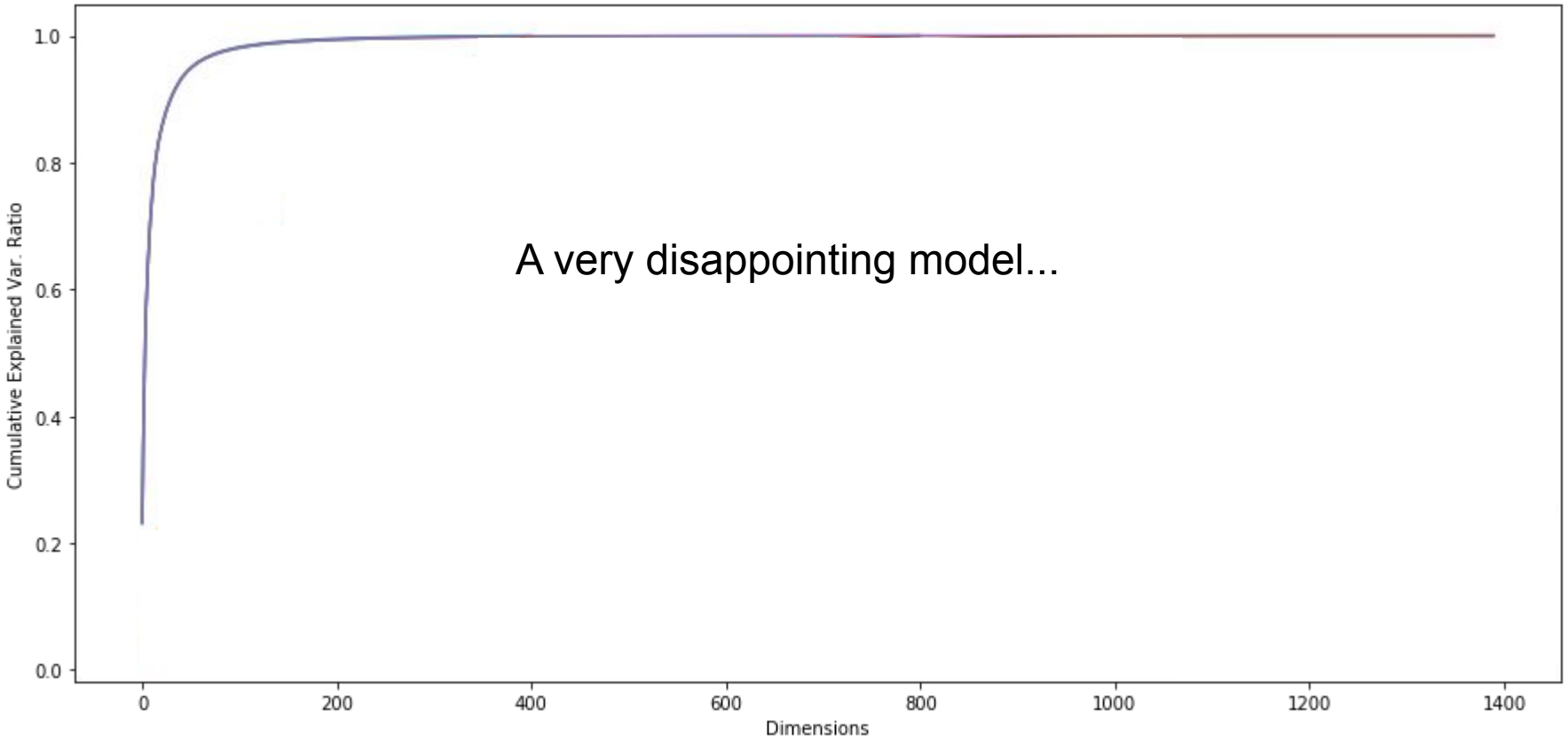
What we found was that the reason the similarities were converging so nicely was because there were only a few latent dimensions what were being used.

i.e. a model with 1000 dimensions really only had ~10-20.

First visualize.

Cumulative explained variance of the principle components

Evaluating Robustness

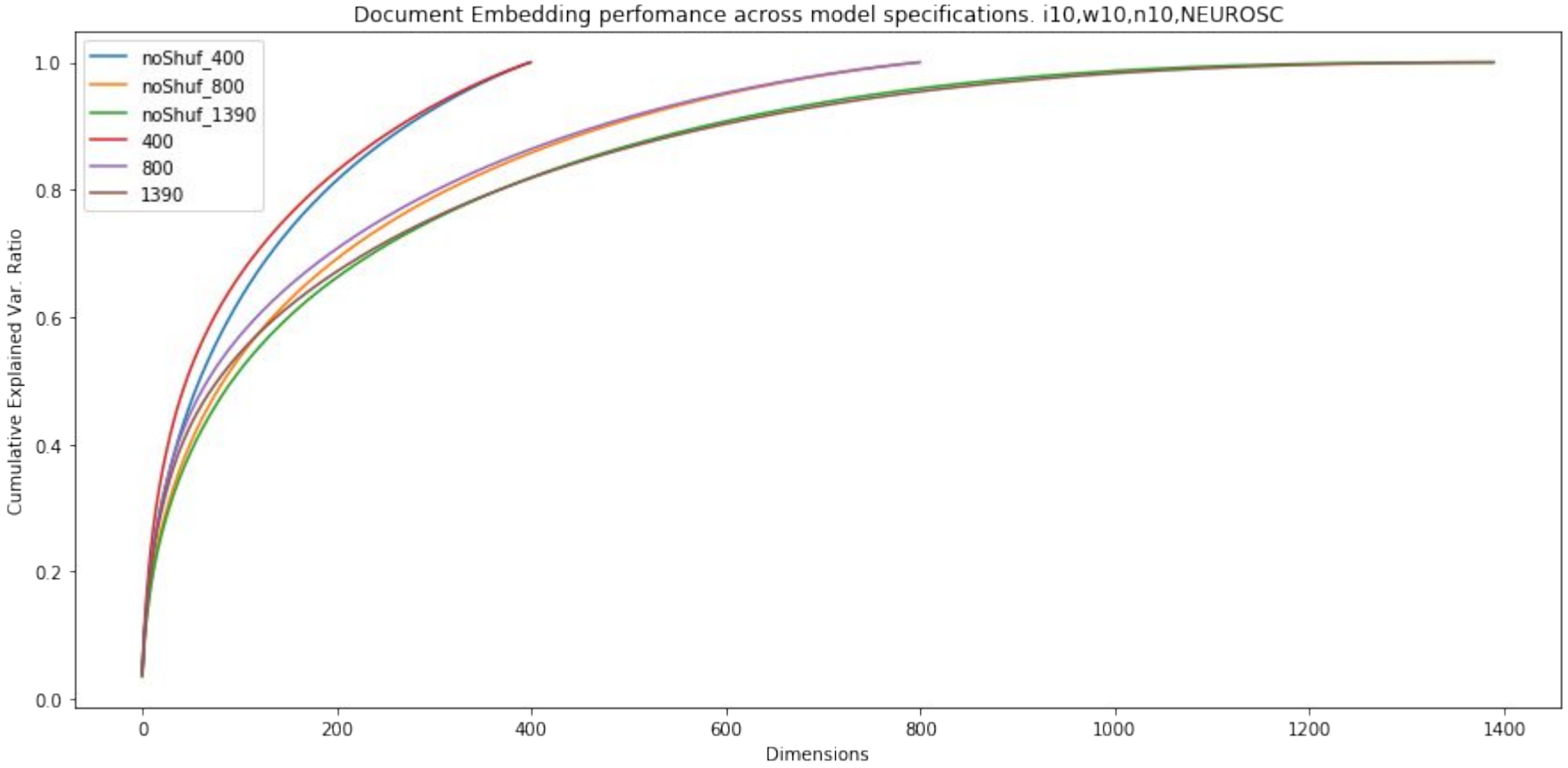


A very disappointing model...

Evaluating Robustness

Back to drawing board on the model parameters.

Evaluating Robustness



Evaluating Robustness - Wrap up

Proposed relatively straightforward approach for evaluating the robustness of topic model output.

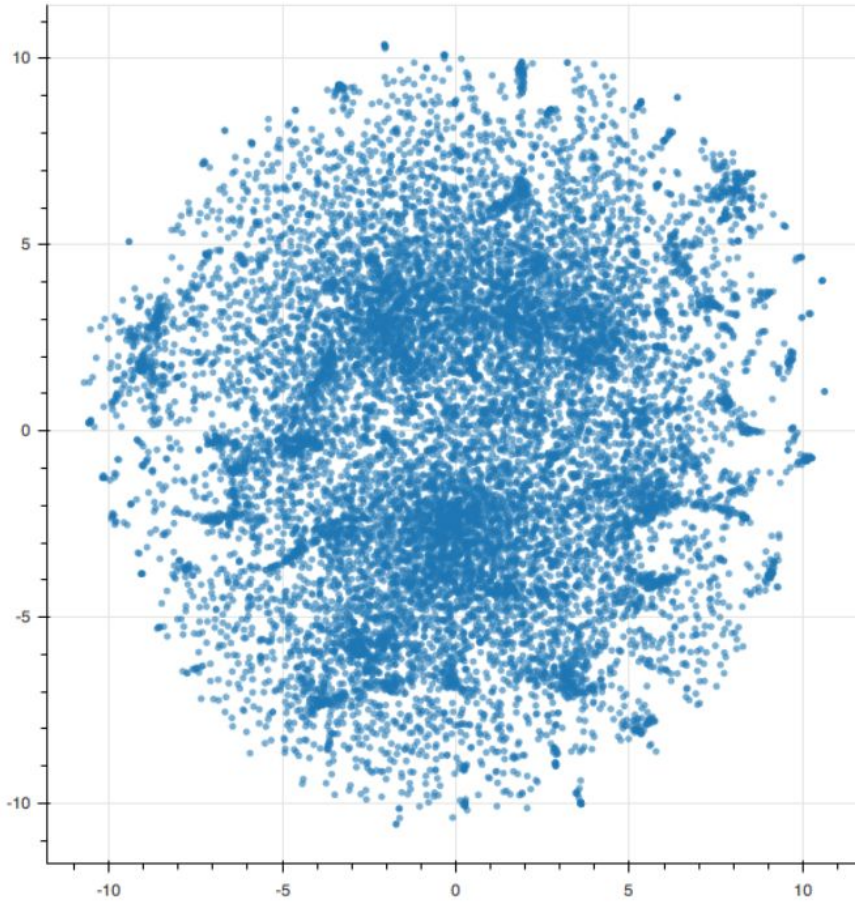
- Based on convergence of pairwise document similarity scores.

Appears as though doc2vec is far more stable than NMF or Latent Dirichlet Allocation.

But right now we don’t really have an explanation for this.

- Could be that doc2vec uses local context, but we have shuffled the words and get very similar results.

Extra: Term Embedding



Extra: Researcher Embedding