Text as Data: Computational Text Analysis

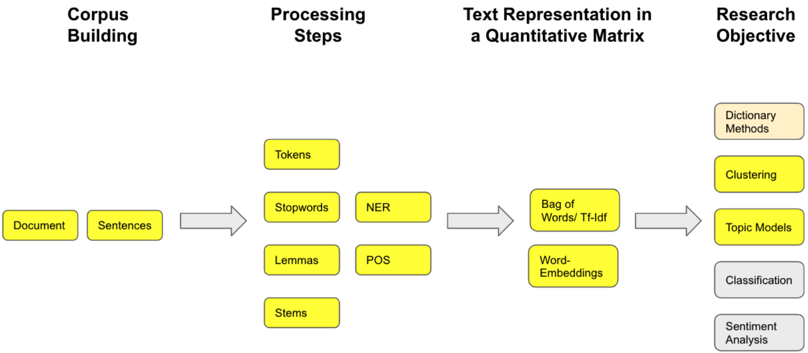
Week 3:
Vector Space Representation and Unsupervised Techniques

Ashrakat Elshehawy

Department of Politics and International Relations,
University of Oxford

May 11, 2021

Overview from Text to Data[1]



[1] Some slides for the session are based on Federico Nanni's course of Computational Text Analysis at U Mannheim

Recap - What have we learned until now?

- Text Pre-Processing

Recap - What have we learned until now?

- Text Pre-Processing
- Stop-words

Recap - What have we learned until now?

- Text Pre-Processing
- Stop-words
- Naive vs NLTK Tokenizer

Recap - What have we learned until now?

- Text Pre-Processing
- Stop-words
- Naive vs NLTK Tokenizer
- Lemmatization and Stemming

Recap - What have we learned until now?

- Text Pre-Processing
- Stop-words
- Naive vs NLTK Tokenizer
- Lemmatization and Stemming
- Pos Tagging and NERF

Recap - What have we learned until now?

- Text Pre-Processing
- Stop-words
- Naive vs NLTK Tokenizer
- Lemmatization and Stemming
- Pos Tagging and NERF
- Dictionaries and Word-counting

Today

- Text Representation & Measuring Text Similarity

Today

- Text Representation & Measuring Text Similarity
  - Bag of Words/Tf-Idf

Today

- Text Representation & Measuring Text Similarity
  - Bag of Words/Tf-Idf
  - Word Embeddings

Today

- Text Representation & Measuring Text Similarity
  - Bag of Words/Tf-Idf
  - Word Embeddings
  - Cosine-Similarity

Today

- Text Representation & Measuring Text Similarity
  - Bag of Words/Tf-Idf
  - Word Embeddings
  - Cosine-Similarity
- Unsupervised Techniques

Today

- Text Representation & Measuring Text Similarity
  - Bag of Words/Tf-Idf
  - Word Embeddings
  - Cosine-Similarity
- Unsupervised Techniques
  - Clustering

Today

- Text Representation & Measuring Text Similarity
  - Bag of Words/Tf-Idf
  - Word Embeddings
  - Cosine-Similarity
- Unsupervised Techniques
  - Clustering
  - Topic-Modelling

Text Representation:

Text to Numbers

Bag of Words

- We have to always first pre-process our text.

Bag of Words

- We have to always first pre-process our text.
    - Tokenization

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case
  - Stopwords, punctuation, and number removal

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case
  - Stopwords, punctuation, and number removal
  - Lemmatization/Stemming

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case
  - Stopwords, punctuation, and number removal
  - Lemmatization/Stemming
- Machines can not process text for algorithms, such as machine learning, in its raw format

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case
  - Stopwords, punctuation, and number removal
  - Lemmatization/Stemming
- Machines can not process text for algorithms, such as machine learning, in its raw format
- They need us to feed in the text into numerical format

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case
  - Stopwords, punctuation, and number removal
  - Lemmatization/Stemming
- Machines can not process text for algorithms, such as machine learning, in its raw format
- They need us to feed in the text into numerical format
- The Bag of Words approach takes some text and count the frequency of the words in that text.

Bag of Words

- We have to always first pre-process our text.
  - Tokenization
  - Lower-case
  - Stopwords, punctuation, and number removal
  - Lemmatization/Stemming
- Machines can not process text for algorithms, such as machine learning, in its raw format
- They need us to feed in the text into numerical format
- The Bag of Words approach takes some text and count the frequency of the words in that text.
- Each word is treated individually, the order of the words does not matter.

Bag of Words[2]

Let's start with the text:

- "John likes to watch movies. Mary likes movies too."
- "John also likes to watch football games."

[2] Example from https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/

Bag of Words[2]

Let's start with the text:

- "John likes to watch movies. Mary likes movies too."
- "John also likes to watch football games."

Pre-processed:

[2] Example from https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/

Bag of Words[2]

Let's start with the text:

- "John likes to watch movies. Mary likes movies too."
- "John also likes to watch football games."

Pre-processed:

- 'john', 'like', 'watch', 'movie', 'mary', 'like', 'movie'
- 'john', 'like', 'watch', 'football', 'game'

[2] Example from https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/

Bag of Words

- We convert a collection of documents/sentences into a numerical matrix

Bag of Words

- We convert a collection of documents/sentences into a numerical matrix
- The goal is to represent each unit of observation as a vector

Bag of Words

- We convert a collection of documents/sentences into a numerical matrix
- The goal is to represent each unit of observation as a vector
- Each unit if observation is a row, each token is a column

	john	like	watch	movie	mary	football	game
sent 1	1	2	1	2	1	0	1
sent 2	1	1	1	0	0	1	1

Bag of Words

- We convert a collection of documents/sentences into a numerical matrix
- The goal is to represent each unit of observation as a vector
- Each unit if observation is a row, each token is a column
- The corresponding (row,column) values being the frequency of occurrence of each word or token in that document.

	john	like	watch	movie	mary	football	game
sent 1	1	2	1	2	1	0	1
sent 2	1	1	1	0	0	1	1

Text-Similarity with Bag-of Words

- Could be a task for classification, topic modeling, sentiment analysis, scaling. There is always this underlining idea of similarity.

Text-Similarity with Bag-of Words

- Could be a task for classification, topic modeling, sentiment analysis, scaling. There is always this underlining idea of similarity.
- We want to know how similar the documents/sentences are to each other

Text-Similarity with Bag-of Words

- Could be a task for classification, topic modeling, sentiment analysis, scaling. There is always this underlining idea of similarity.
- We want to know how similar the documents/sentences are to each other

	Sent 1	Sent 2
art	1	1
commit	1	0
Angela_Merkel	0	1
government	1	0
Theresa_May	1	0
intend	0	1
say	1	1
she	0	1
support	1	1
to be	1	0

Tf-Idf

Instead of using the raw frequency, you can normalize it giving less weight to very frequent words

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Tf-Idf

	Sent 1	Sent 2
art	0.8	0.8
commit	0.7	0
Angela_Merkel	0	0.9
government	0.8	0
Theresa_May	0.8	0
intend	0	0.7
say	0.4	0.4
she	0	0.1
support	0.6	0.6
to be	0.1	0

Benefits and Drawbacks

Pros:

- Low computational cost

Benefits and Drawbacks

Pros:

- Low computational cost
- Good for classification, clustering

Benefits and Drawbacks

Pros:

- Low computational cost
- Good for classification, clustering

Cons:

Benefits and Drawbacks

Pros:

- Low computational cost
- Good for classification, clustering

Cons:

- No theory behind these models

Benefits and Drawbacks

Pros:

- Low computational cost
- Good for classification, clustering

Cons:

- No theory behind these models
- Sparse vectors (vectors with a lot of zero scores)

Benefits and Drawbacks

Pros:

- Low computational cost
- Good for classification, clustering

Cons:

- No theory behind these models
- Sparse vectors (vectors with a lot of zero scores)
- Discarding word order and meaning $\rightarrow$ example ("this is interesting" vs "is this interesting")

Word-Embeddings

"A word embedding is a vector, and is a set of estimated weights from a neural network model" (A. Spirling).

Word-Embeddings

"A word embedding is a vector, and is a set of estimated weights from a neural network model" (A. Spirling).

So how do we capture a meaning of a word? bigskip

Word-Embeddings

"A word embedding is a vector, and is a set of estimated weights from a neural network model" (A. Spirling).

So how do we capture a meaning of a word? bigskip Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

Word-Embeddings

"A word embedding is a vector, and is a set of estimated weights from a neural network model" (A. Spirling).

So how do we capture a meaning of a word? bigskip Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

How do they help us?

Word-Embeddings

"A word embedding is a vector, and is a set of estimated weights from a neural network model" (A. Spirling).

So how do we capture a meaning of a word? bigskip Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

How do they help us?

- Identify similarities between words

Word-Embeddings

"A word embedding is a vector, and is a set of estimated weights from a neural network model" (A. Spirling).

So how do we capture a meaning of a word? bigskip Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

How do they help us?

- Identify similarities between words
- By checking words in the vector space, word-embedding models can check analogies

Word-Embeddings

In 2013, Tomas Mikolov proposed to learn word vectors using a neural network with a single hidden layer (F Nanni).

Word-Embeddings

In 2013, Tomas Mikolov proposed to learn word vectors using a neural network with a single hidden layer (F Nanni).

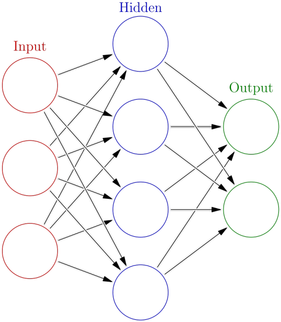
What is a neural net?

Word-Embeddings

In 2013, Tomas Mikolov proposed to learn word vectors using a neural network with a single hidden layer (F Nanni).

What is a neural net?

A computing system loosely inspired by the structure of the brain.



Wikipedia Link

Word-Embeddings

Train the neural network for two different tasks:

Word-Embeddings

Train the neural network for two different tasks:

- CBOW: predicting the word, given the context

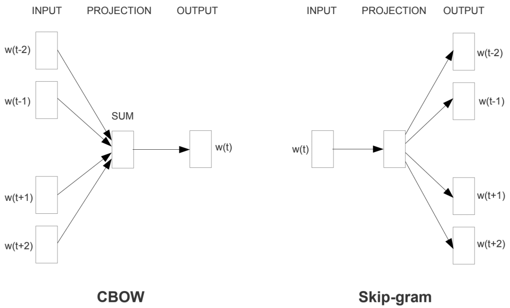


Image source: Christian Perone Website

Word-Embeddings

Train the neural network for two different tasks:

- CBOW: predicting the word, given the context
- Skip-gram: predict the context, given a word

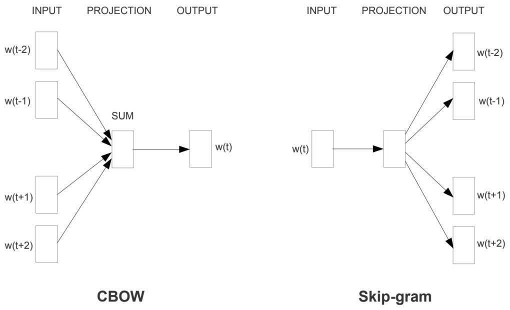


Image source: Christian Perone Website

Word-Embeddings

Word2Vec:

- Mikolov et al. (2013) developed Word2Vec while working at Google Brain.

Word-Embeddings

Word2Vec:

- Mikolov et al. (2013) developed Word2Vec while working at Google Brain.

- It can produce word embeddings, given a corpus.

Word-Embeddings

Word2Vec:

- Mikolov et al. (2013) developed Word2Vec while working at Google Brain.

- It can produce word embeddings, given a corpus.

- Each word is represented by a real-valued vector, often tens or hundreds of dimensions

Word-Embeddings

Word2Vec:

- Mikolov et al. (2013) developed Word2Vec while working at Google Brain.

- It can produce word embeddings, given a corpus.

- Each word is represented by a real-valued vector, often tens or hundreds of dimensions

```
array([ 0.24844994, -0.2488279 ,  0.28766018, -0.06125903, -0.39593282,
        0.03418331,  0.17195873,  0.14292698, -0.27260116, -0.03628655,
        0.016991  ,  0.2809293 , -0.2928955 , -0.23852736, -0.33930627,
       -0.5453123 ,  0.1066637 ,  0.05541009,  0.01136546, -0.19054835,
       -0.12419918, -0.01751846,  0.01711187, -0.025536  ,  0.07011069,
        0.512462  , -0.0982796 , -0.08487804, -0.02550475, -0.23541924,
       -0.03985457,  0.08368545,  0.19705558,  0.03259188,  0.11399814,
        0.10220459,  0.49121043, -0.0096594 ,  0.2938598 ,  0.18612786,
        0.12146058,  0.26321754, -0.18536666,  0.21491598, -0.19445091,
        0.04173627, -0.01773   ,  0.26945433,  0.12963997, -0.07131842,
        0.14261793,  0.34846178, -0.154857  , -0.1372897 ,  0.00520176,
        0.33702272,  0.35870683,  0.26573738,  0.24198972, -0.09749658,
        0.02182623,  0.25890318,  0.12668978, -0.04532094,  0.48377496,
       -0.34790802, -0.11821937, -0.26788604, -0.13667464,  0.20555115,
        0.16379716, -0.26927355,  0.11574815, -0.7150321 ,  0.19101486,
       -0.05553193, -0.1575609 ,  0.1934963 ,  0.05073268,  0.05009224,
       -0.52024734, -0.14362402, -0.13356815, -0.51892006,  0.19463988,
       -0.07593681,  0.05317167, -0.25996113,  0.01347924,  0.15779038,
        0.00995404,  0.11217199,  0.4228523 ,  0.16897605, -0.1499024 ,
```

For visualization check this `https://projector.tensorflow.org/`

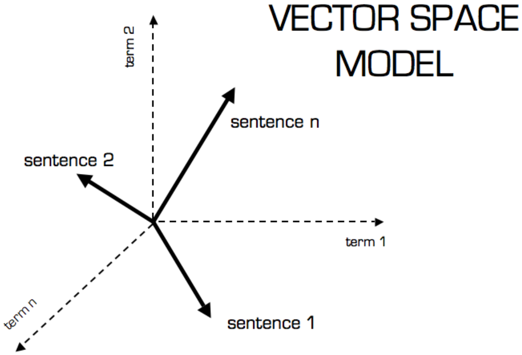
Cosine-Similarity

- Cosine similarity calculates similarity by measuring the cosine of angle between two vectors in a multi-dimensional space.

Cosine-Similarity

- Cosine similarity calculates similarity by measuring the cosine of angle between two vectors in a multi-dimensional space.

- The smaller the angle the higher the cosine similarity



Unsupervised Techniques

Clustering

- This method aims to partition n observations (i.e., documents) into groups of clusters.

Figure: Source: Analytics Vidha

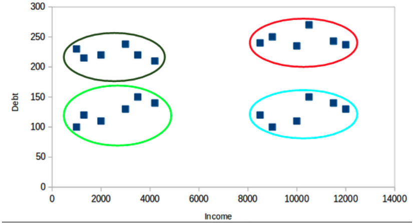


Figure: Source: Analytics Vidha

Clustering

- This method aims to partition n observations (i.e., documents) into groups of clusters.
- **K-Means**: Each observation belongs to the cluster with the nearest mean (uses cosine-similarity)

Figure: Source: Analytics Vidha

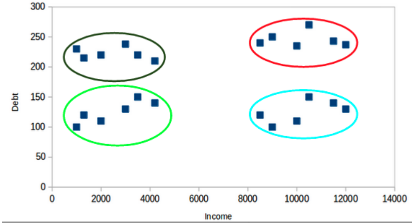


Figure: Source: Analytics Vidha

Clustering

- This method aims to partition n observations (i.e., documents) into groups of clusters.
- **K-Means**: Each observation belongs to the cluster with the nearest mean (uses cosine-similarity)
- You do not know classes in advance.

Figure: Source: Analytics Vidha

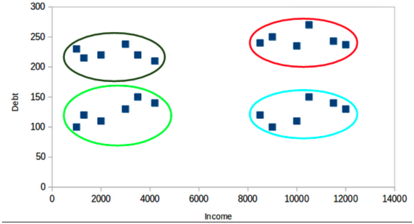


Figure: Source: Analytics Vidha

Clustering

- This method aims to partition n observations (i.e., documents) into groups of clusters.
- **K-Means**: Each observation belongs to the cluster with the nearest mean (uses cosine-similarity)
- You do not know classes in advance.
- You have to choose the number of clusters!

Figure: Source: Analytics Vidha

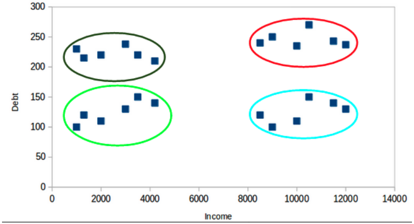


Figure: Source: Analytics Vidha

Clustering

Issues:

- Very coarse-grained

Clustering

Issues:

- Very coarse-grained
- Numbers of clusters have to be assigned in advance

Clustering

Issues:

- Very coarse-grained
- Numbers of clusters have to be assigned in advance
- Difficult to evaluate

Topic Modelling[3]

- Instead of assigning each document to a single cluster, we identify the underlying topics of each document and group them.

[3] Explanation partially on C. Bail's Duke SICC, F. Nanni 2018 at Uni Mannheim, A. Spirling 2021 at NYU

Topic Modelling[3]

- Instead of assigning each document to a single cluster, we identify the underlying topics of each document and group them.

- In k-means clustering, each observation can be assigned to only one cluster.

[3] Explanation partially on C. Bail's Duke SICC, F. Nanni 2018 at Uni Mannheim, A. Spirling 2021 at NYU

Topic Modelling[3]

- Instead of assigning each document to a single cluster, we identify the underlying topics of each document and group them.

- In k-means clustering, each observation can be assigned to only one cluster.

- Topic models, however, are mixture models. Each observation/doc is assigned a probability of belonging to a latent theme or "topic."

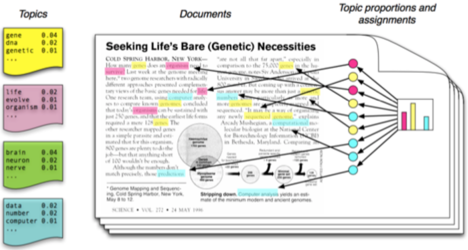


Image source: Blei, David M. "Probabilistic topic models." Communications of the ACM 55.4 (2012): 77-84.

[3] Explanation partially on C. Bail's Duke SICC, F. Nanni 2018 at Uni Mannheim, A. Spirling 2021 at NYU

Topic-Modelling

LDA: Latent Drichlet Allocation

- A word in a document is likely to belong to the same topic as the other words of that document

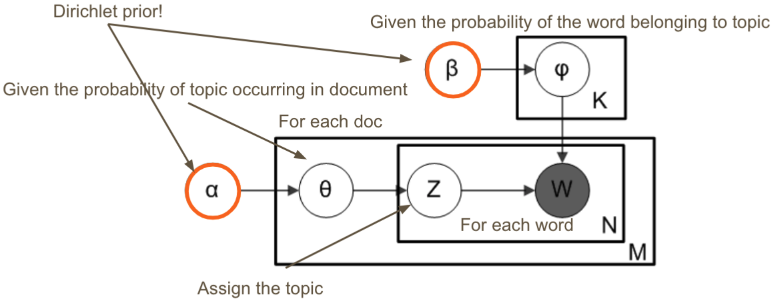


Image source: Federico Nanni 2018

Topic-Modelling

LDA: Latent Drichlet Allocation

- A word in a document is likely to belong to the same topic as the other words of that document
- The distribution of topics in documents and distribution of words in topics have a Dirichlet prior

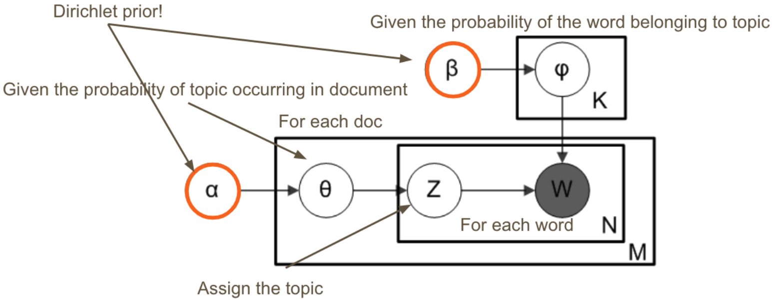


Image source: Federico Nanni 2018

Topic-Modelling

LDA: Latent Drichlet Allocation

- A word in a document is likely to belong to the same topic as the other words of that document
- The distribution of topics in documents and distribution of words in topics have a Dirichlet prior

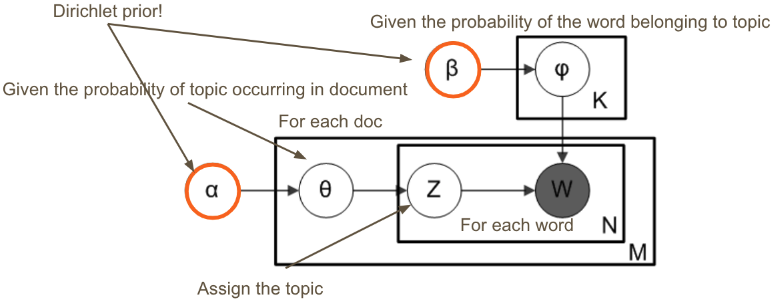


Image source: Federico Nanni 2018

Topic-Modelling

LDA: Latent Drichlet Allocation (Gibbs Sampling)

- Initialize topic assignments randomly
- **For each iteration**
  - **For each document**
    - **For each word re-assign topic to word, given**:
    - *all other words in the doc and their topic-assignment (dirichlet prior)*
    - *all other occurrences of the same word in other docs (dirichlet prior)*

Topic-Modelling

Output: First of all, we need to know that we do not get topic-labels.

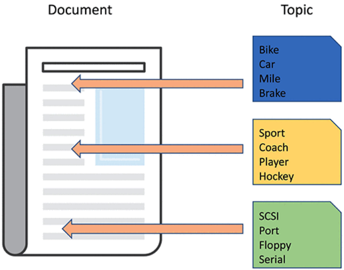


Image source: Amazon Blogs

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic
- Substitute 1 of them with a top word from another topic

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic
- Substitute 1 of them with a top word from another topic
- Shuffle

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic
- Substitute 1 of them with a top word from another topic
- Shuffle
- Ask coder to pick the intruder

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic
- Substitute 1 of them with a top word from another topic
- Shuffle
- Ask coder to pick the intruder

Topic-Intrusion Task:

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic
- Substitute 1 of them with a top word from another topic
- Shuffle
- Ask coder to pick the intruder

Topic-Intrusion Task:

- Present coders with 4 topics, one of them is an intruder topic

Topic-Modelling

One way of Evaluation - ask humans to evaluate:

Word-Intrusion Task:

- Take the top k (usually 5) words in a topic
- Substitute 1 of them with a top word from another topic
- Shuffle
- Ask coder to pick the intruder

Topic-Intrusion Task:

- Present coders with 4 topics, one of them is an intruder topic
- If they all guess same intruder/outlier topic, topics are relevant.