A CLOUD GURU

Text Feature Engineering

Brock Tubre
INSTRUCTOR

Feature Engineering

Transforming attributes within our data to make them more useful within our model for the problem at hand. Feature engineering is often compared to an art.

Text Feature Engineering

Text Feature Engineering

Transforming text within our data so Machine Learning algorithms can better analyze it.

Splitting text into bite size pieces.

Text Feature Engineering

A CLOUD GURU

AAPS PharmSciTech, Vol. 17, No. 1, February 2016 (© 2016)
DOI: 10.1208/s12249-016-0486-2

Editorial

Theme: Pharmaceutical Thermal Processing
Guest Editors: Feng Zhang and Michael A. Repka

Pharmaceutical Thermal Processing

Feng Zhang[1,3] and Michael A. Repka[2]

published online 9 February 2016

Thermal processing has flourished as a novel technology to prepare a wide range of drug delivery systems over the past decade. Thermal processing is built on the principle of efficient conversion of the mechanical energy into the thermal energy of the formulation matrix. Single and twin screw extruders are the most commonly employed thermal processing equipment. Kinetisol® Dispersing technology, an innovative thermal process, was invented recently. This theme issue of AAPS PharmSciTech provides a broad overview of (a) the history and recent advancement of thermal processing, (b) scientific approaches to develop drug delivery systems using thermal processing, and (c) examples of a wide range of drug delivery systems prepared using thermal processing.

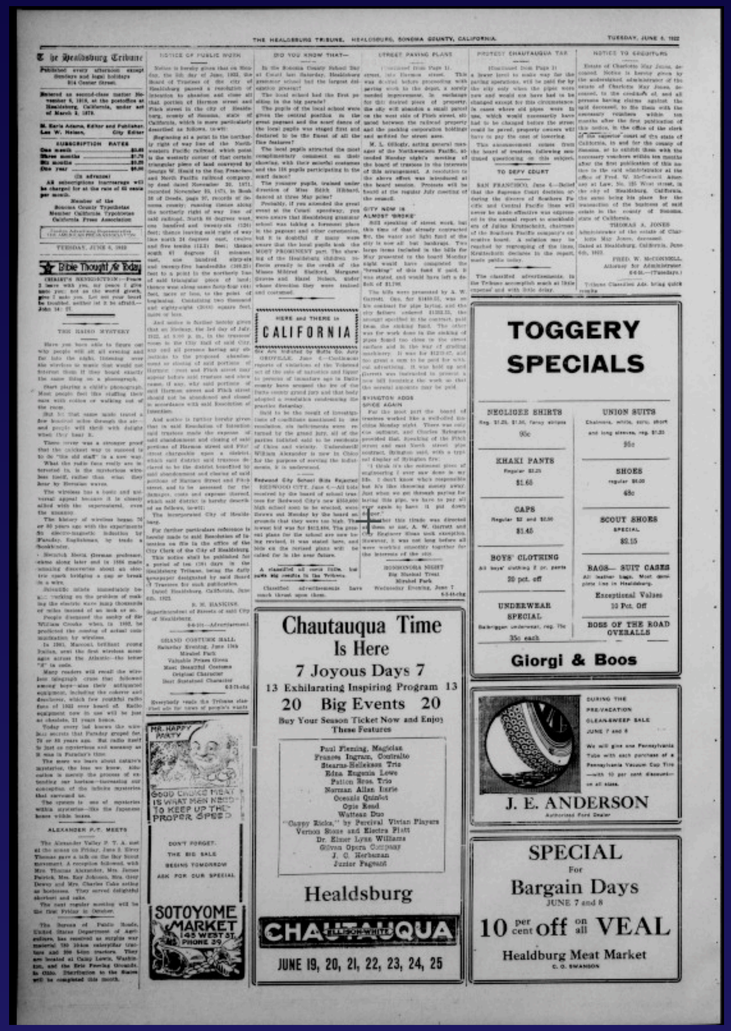
This issue of AAPS PharmSciTech offers three review articles. Martin provided a historical and technical review of a twin screw extruder as a continuous mixer for thermal processing in both the plastic and pharmaceutical industry from an equipment engineer's perspective (1). From his personal experience, Martin believed that the evolution of twin screw extrusion-based thermal processing in the pharmaceutical industry resembles what occurred for the plastics industry in 1980s. Patil et al. reviewed the application of melt extrusion in the production of diverse pharmaceutical delivery systems (2). The authors emphasized the critical role of the quality-by-design approach and process analytical technology in pharmaceutical melt extrusion monitoring and control. LaFountaine et al. discussed a number of formulation and process-driven strategies to enable thermal processing of challenging compositions (3). These included the use of traditional plasticizers and surfactants, temporary plasticizers utilizing sub- or supercritical carbon dioxide, designer polymers, and KinetiSol® Dispersing technology.

This theme issue also contains 15 original research articles. Bhagurkar et al. proposes the use of melt extrusion processing for the preparation of a polyethylene glycol base ointment (4). A modified screw design was used and parameters such as feeding rate, barrel temperature, and screw speed were optimized to obtain a uniform product. Zhang successfully applied a melt extrusion process to prepare Eudragit FS-based granules for colon-targeting drug delivery (5). Maniruzzaman et al. investigated the effect of novel melt-extruded polymer and lipid formulations on the dissolution rates of indomethacin (6). A synergy between the polymer and lipid in enhancing the drug release was reported. Ye et al. demonstrated the feasibility of producing nanocrystal solid dispersions by conjugating high-pressure homogenization and melt extrusion processes (7). Jedinger et al. developed cornstarch-based flexible deformable multiple-unit pellets with abuse deterrence properties using a twin screw extruder (8). Lamm and his colleagues characterized the phase behavior and morphology of melt-extruded amorphous solid dispersions comprised of copovidone and TPGS 1000 using atomic force microscopy and differential scanning calorimetry (9). Both the processing parameters including screw speed and cooling method as well as the compositional effects on the resulting melt extrudate were studied. Agrawal et al. presented a case study of systematic development and scale up of an amorphous solid dispersion dosage to the clinical scale using less than 500 g drug (10). The melt extrusion process was scaled up from lab scale to clinical scale using a volumetric scale up approach and scale independent-specific energy parameter. In a separate study, Agrawal et al. evaluated the influence of extra-granular components on dissolution of tablets containing melt-extruded amorphous solid dispersion granules (11). To achieve rapid disintegration and acceptable tensile strength, the authors found that the extra-granular composition was also dependent on both the type of polymer the drug-polymer interaction. LaFountaine et al. demonstrated that Kinetisol® Dispersing technology produced amorphous dispersions at higher drug loads than could be prepared by melt extrusion, as well as with higher molecular weight polymers that were not processable by melt extrusion (12). Li et al. applied Flory–Huggins and temperature–composition phase diagrams to not only understand drug–polymer miscibility behavior but also rationalize the selection of important processing parameters for melt extrusion to ensure miscibility of drug and polymer (13). Five research articles are focused on the application of novel excipients in melt extrusion process. Brough et al. enabled polyvinyl alcohol as the polymeric carrier for amorphous solid

[1] College of Pharmacy, University of Texas at Austin, 2409 University Ave, Austin, Texas 78712, USA.
[2] Department of Pharmaceutics and Drug Delivery, School of Pharmacy, University of Mississippi, University, Oxford, Mississippi 38677, USA.
[3] To whom correspondence should be addressed. (e-mail: feng.zhang@austin.utexas.edu)


```
Dialogue: d92-1
Number of utterances files: 54
Length of dialogue: 189.448858
Estimated number of turns: 32

utt1  : s:  hello <sil> can I help you
utt2  : u:  yeah I want t- I want to determine the maximum number of boxcars of
            oranges <sil> that I can get to Bath <sil> by seven a.m. <sil>
            tomorrow morning
utt3  :     so <brth> hm <sil>
            so I guess all the boxcars will have to go through oran- <sil>
            through Corning because that's where the orange juice <brth>
            orange factory is
utt4  :     so from Corning to Bath how far is that
utt5  : s:  two hours
utt6  : u:  and it's gonna take us also an hour to load <sil> boxcars right
utt7  : s:  right + +
utt8  : u:  + okay + so <sil> hm so <sil> every trip will take at least <sil>
            three hours <sil> then
utt9  :     um
utt10 : s:  right we can unload any amount of cargo onto a train in one hour
utt11 :     so we can + <sil> do a maximum of three + boxcars in an hour
utt12 : u:  + right <sil> okay +
utt13 :     okay <sil> so I guess one thing we can do oh <brth> so <brth>
            I guess one thing is that we should see how many boxcars we can actually
            get to Corning in four hours
utt14 :     um how far is it from Avon to Bath <sil> to Corning
utt15 : s:  <click> <brth> that's six hours it's + shorter + through Dansville
utt16 : u:  + okay +
utt17 :     from Avon oh no but the thing is is that I was thinking if s- <sil>
            I was wondering if we could actually pick up those two boxcars which are
            at <sil> + Bath +
utt18 : s:  + mm +
utt19 : u:  but I think that we can't even put them in <sil> to <brth>
            that they won't even come in to play
utt20 :     unless how far is it from Elmira <brth> to Bath
utt21 : s:  two hours
utt22 : u:  oh really <sil> so then w- we could actually take <brth>
            like engine E two
utt23 :     have it go to bath <sil> pick up those two box cars there
utt24 :     take it to Corn- how long from Bath to Corning an hour I guess
utt25 : s:  two hours
utt26 : u:  oh wait a second I thought <sil> well from from like Elmira <brth>
            through to Corning to Bath is how many hours
utt27 : s:  four hours + +
utt28 : u:  + oh + that's four hours so wa- we're like screwed as far as tho- those
            two box cars at Bath go
utt29 :     like I don- I don't think we s- so to get the maximum number of box cars
            of oranges <brth>
            to bath by seven am I don't think we can even use those two at bath
utt30 :     because there <sil> we because we can't get an engine <brth>
            to Bath in time
utt31 : s:  that's right
utt32 : u:  okay <brth> so we have the three boxcars at <sil> Dansville
utt33 :     so how far is it from Avon to Danville
utt34 : s:  three hours
```



THE HEALDSBURG TRIBUNE, HEALDSBURG, SONOMA COUNTY, CALIFORNIA. TUESDAY, JUNE 6, 1922

The Healdsburg Tribune

Bible Thought & Today

CALIFORNIA

Chautauqua Time Is Here
7 Joyous Days 7
13 Exhilarating Inspiring Program 13
20 Big Events 20
Buy Your Season Ticket Now and Enjoy These Features
Healdsburg
CHAUTAUQUA
JUNE 19, 20, 21, 22, 23, 24, 25

TOGGERY SPECIALS
Giorgi & Boos

J. E. ANDERSON

SPECIAL For Bargain Days JUNE 7 and 8
10 per cent off on all VEAL
Healdsburg Meat Market
C. O. SWANSON

Text Feature Engineering

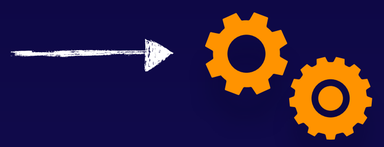
Example:

Raw Text: { “123 Main Street, Seattle, WA 98101” }

Text Feature Engineering

Example:

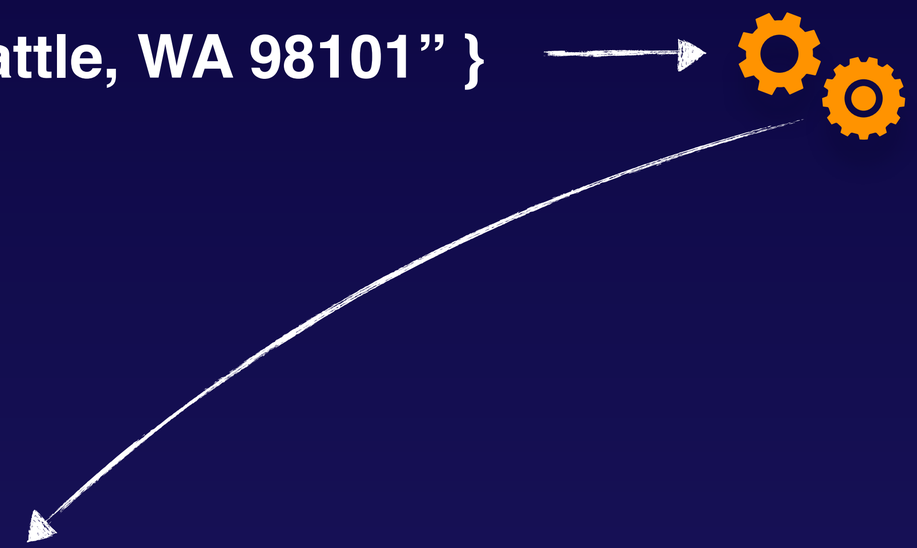
Raw Text: { “123 Main Street, Seattle, WA 98101” }



Text Feature Engineering

Example:

Raw Text: { “123 Main Street, Seattle, WA 98101” }



Address	City	State	Zip
123 Main Street	Seattle	WA	98101

Bag-of-Words

Bag-of-Words

Tokenizes raw text and creates a statistical representation of the text.

Breaks up text by white space into single words.

Bag-of-Words

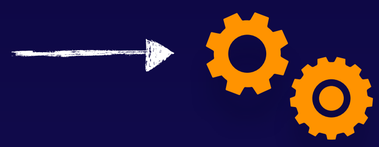
Example:

Raw Text: { “he is a jedi and he will save us” }

Bag-of-Words

Example:

Raw Text: { “he is a jedi and he will save us” }

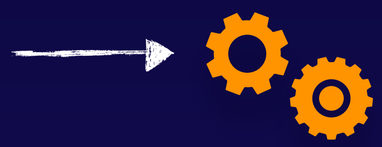


Bag-of-Words

Bag-of-Words

Example:

Raw Text: { “he is a jedi and he will save us” }



Bag-of-Words

{ “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us

Bag-of-Words

Example:

Raw Text: { “he is a jedi and he will save us” }

Bag-of-Words

ID	word	count
1	he	2
2	is	1
3	a	1
4	jedi	1
5	and	1
6	will	1
7	save	1
8	us	1

{ “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us

N-Gram

N-Gram

An extension of Bag-of-Words which produces groups of words of *n* size.

Breaks up text by white space into groups of words.

N-Gram Example (1-gram)

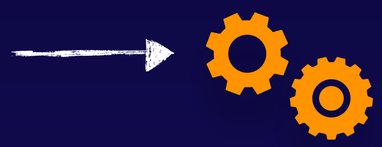
Example:

Raw Text: { “he is a jedi and he will save us” }

N-Gram Example (1-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

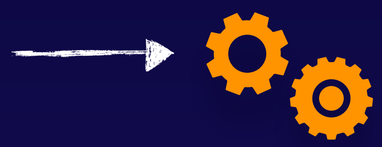


N-gram, size = 1

N-Gram Example (1-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }



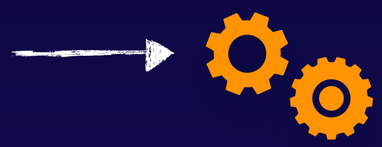
N-gram, size = 1

{ “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

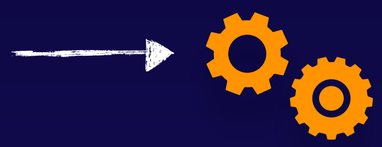


N-gram, size = 2

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }



N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

N-gram, size = 2

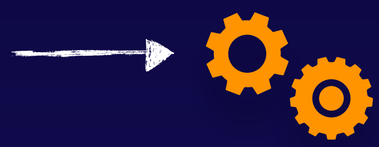
“sliding window”
n = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }



N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”,
“he will”, “will save”, “save us”, “he”, “is”, “a”,
“jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

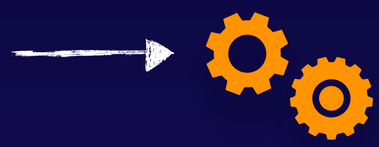
N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will” “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }



N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will” “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” }

N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

N-Gram Example (2-gram)

Example:

Raw Text: { “he is a jedi and he will save us” } → 

N-gram, size = 2

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”, “he will”, “will save”, “save us”, “he”, “is”, “a”, “jedi”, “and”, “he”, “will”, “save”, “us”}

Unigrams, Bigrams, Trigrams...

unigram = 1 word tokens

bigram = 2 word tokens

trigram = 3 word tokens

...

Orthogonal Sparse Bigram (OSB)

Orthogonal Sparse Bigram (OSB)

Creates groups of words of size *n* and outputs every pair of words that includes the first word.

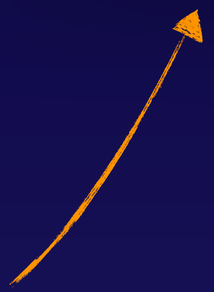
Creates groups of words that always include the first word.

Orthogonal Sparse Bigram (OSB)

Orthogonal Sparse Bigram

Orthogonal Sparse Bigram (OSB)

Orthogonal Sparse Bigram



When two things are
independent of each other

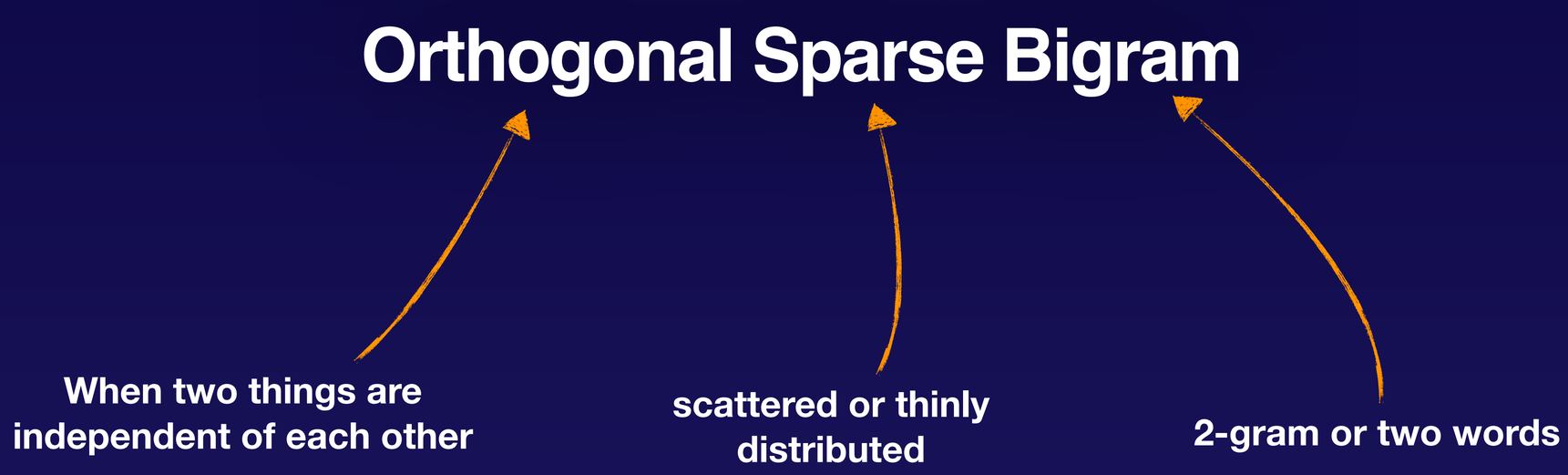
Orthogonal Sparse Bigram (OSB)

Orthogonal Sparse Bigram

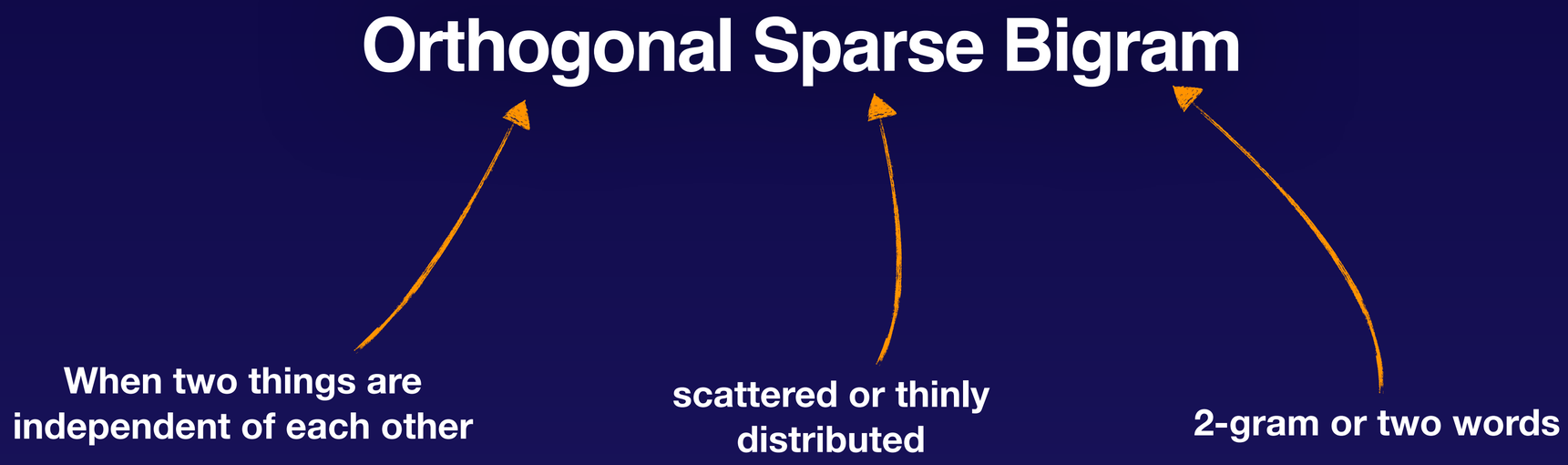
When two things are independent of each other

scattered or thinly distributed

Orthogonal Sparse Bigram (OSB)



Orthogonal Sparse Bigram (OSB)



OSB is not “better” or “worse” than N-Gram, it’s just another technique to use…

Orthogonal Sparse Bigram (OSB)

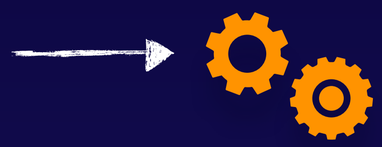
Example:

Raw Text: { “he is a jedi and he will save us” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }



OSB, size = 4

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

“sliding window”
n = 4

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

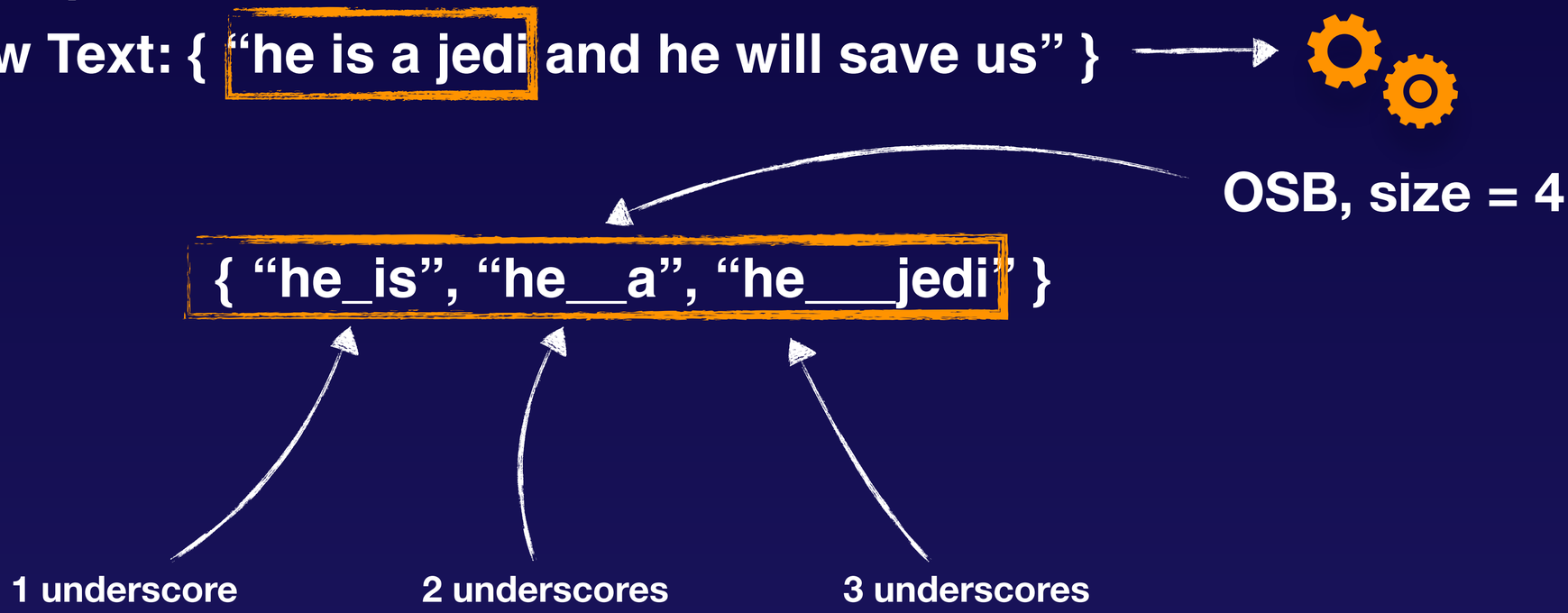
{ “he_is”, “he__a”, “he___jedi” }

underscore = white space

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }



Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }

{ “is_a”, “is__jedi”, “is___and” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }

{ “is_a”, “is__jedi”, “is___and” }

{ “a_jedi”, “a__and”, “a___he” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }

{ “is_a”, “is__jedi”, “is___and” }

{ “a_jedi”, “a__and”, “a___he” }

{ “jedi_and”, “jedi__he”, “jedi___will” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }

{ “is_a”, “is__jedi”, “is___and” }

{ “a_jedi”, “a__and”, “a___he” }

{ “jedi_and”, “jedi__he”, “jedi___will” }

{ “and_he”, “and__will”, “and___save” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }
{ “is_a”, “is__jedi”, “is___and” }
{ “a_jedi”, “a__and”, “a___he” }
{ “jedi_and”, “jedi__he”, “jedi___will” }
{ “and_he”, “and__will”, “and___save” }
{ “he_will”, “he__save”, “he___us” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }
{ “is_a”, “is__jedi”, “is___and” }
{ “a_jedi”, “a__and”, “a___he” }
{ “jedi_and”, “jedi__he”, “jedi___will” }
{ “and_he”, “and__will”, “and___save” }
{ “he_will”, “he__save”, “he___us” }
{ “will_save”, “will__us” }

Orthogonal Sparse Bigram (OSB)

Example:

Raw Text: { “he is a jedi and he will save us” }

OSB, size = 4

{ “he_is”, “he__a”, “he___jedi” }
{ “is_a”, “is__jedi”, “is___and” }
{ “a_jedi”, “a__and”, “a___he” }
{ “jedi_and”, “jedi__he”, “jedi___will” }
{ “and_he”, “and__will”, “and___save” }
{ “he_will”, “he__save”, “he___us” }
{ “will_save”, “will__us” }
{ “save_us” }

Tf-idf

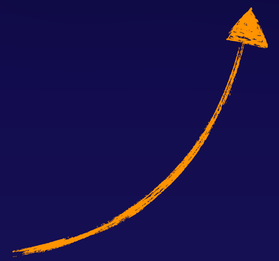
Term Frequency - Inverse Document Frequency (tf-idf)

Represents how important a word or words are to a given set of text by providing appropriate weights to terms that are common and less common in the text.

Shows us the popularity of a word or words in text data by making common words like “the” or “and” less important.

Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency

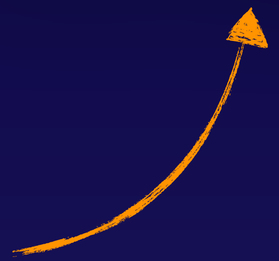


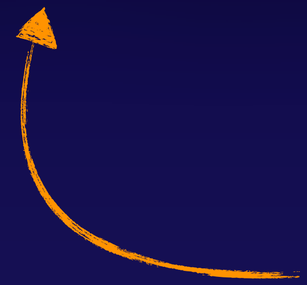
How frequent does a word appear.

Tf-idf

Term Frequency - Inverse Document Frequency

How frequent does a word appear.





Number of documents in which terms occur.

Tf-idf

Term Frequency - Inverse Document Frequency

How frequent does a word appear.

Makes common words less meaningful.

Number of documents in which terms occur.

Tf-idf

Example:

Tf-idf

Example:

tf-idf

Document 1

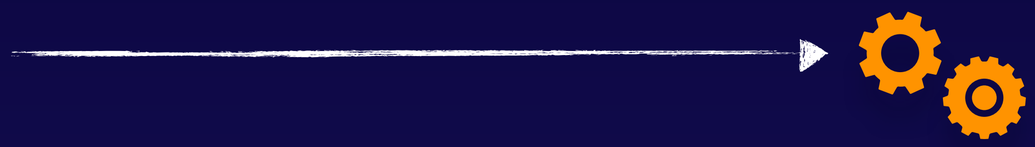
word	count
the	3
force	1
Luke	1
Skywalker	1
a	2

Document 2

word	count
the	2
jedi	1
a	1
empire	1

Tf-idf

Example:



tf-idf

Document 1

word	count
the	3
force	1
Luke	1
Skywalker	1
a	2

Document 2

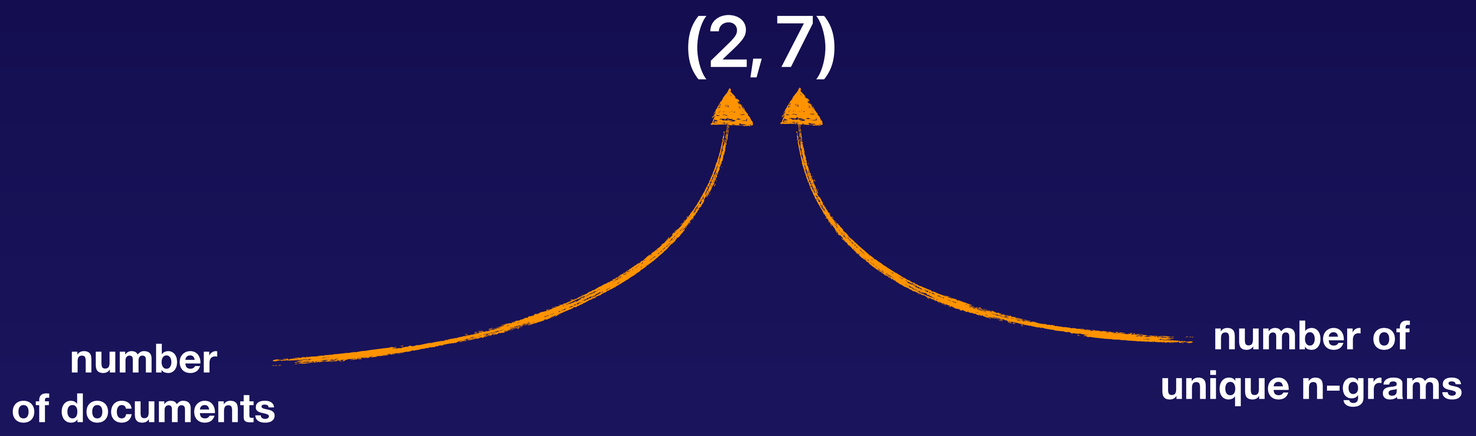
word	count
the	2
jedi	1
a	1
empire	1

Since the tokens “the” and “a” showed up in BOTH documents many times, these are deemed as less important.

Tf-idf Vectorizer

(number of documents, number of unique n-grams)

Tf-idf Vectorizer



Use Cases

Problem	Transformation	Why
Matching phrases in spam emails	N-Gram	Easier to compare whole phrases like “click here now” or “you’re a winner”.
Determining the subject matter of multiple PDFs	Tf-idf Orthogonal Sparse Bigram	Filter less important words in PDFs. Find common word combinations repeated throughout PDFs.

Text Feature Engineering

- **Remove Punctuation**

  Sometimes removing punctuation is a good idea if we do not need them.

- **Lowercase Transformation**

  Using lowercase transformation can help standardize raw text.

Removing Punctuation

Example:

Raw Text: { “Help me, Obi-Wan Kenobi. You’re my only hope.” }

Removing Punctuation

Example:

Raw Text: { “Help me, Obi-Wan Kenobi. You’re my only hope.” }

remove punctuation

Removing Punctuation

Example:

Raw Text: { “Help me, Obi-Wan Kenobi. You’re my only hope.” }

remove punctuation

{ “Help”, “me”, “Obi-Wan”, “Kenobi”, “You’re”, “my”, “only”, “hope” }

Removing Punctuation

Example:

Raw Text: { “Help me, Obi-Wan Kenobi. You’re my only hope.” }

remove punctuation

{ “Help”, “me”, “Obi-Wan”, “Kenobi”, “You’re”, “my”, “only”, “hope” }

Removing Punctuation

Example:

Raw Text: { “Help me, Obi-Wan Kenobi. You’re my only hope.” }

remove punctuation

{ “Help”, “me”, “Obi-Wan”, “Kenobi”, “You’re”, “my”, “only”, “hope” }

Lowercase Transformation

Example:

Raw Text: { “I Find Your Lack of Faith Disturbing” }

Lowercase Transformation

Example:

Raw Text: { “I Find Your Lack of Faith Disturbing” } → 

lowercase

Lowercase Transformation

Example:

Raw Text: { “I Find Your Lack of Faith Disturbing” }

lowercase

{ “i find your lack of faith disturbing” }

Cartesian Product

Cartesian Product Transformation

Creates a new feature from the combination of two or more text or categorical values.

Combining sets of words together.

Cartesian Product

Cartesian Product

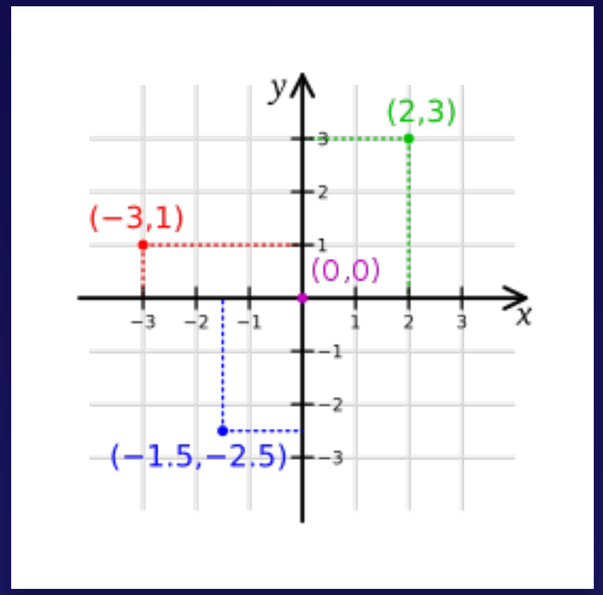
Cartesian Product

René Descartes
c.1596 – c.1650
Latinized: Renatus Cartesius

Cartesian Product

Cartesian Product

René Descartes
c.1596 – c.1650
Latinized: Renatus Cartesius



Cartesian Product

Cartesian Product

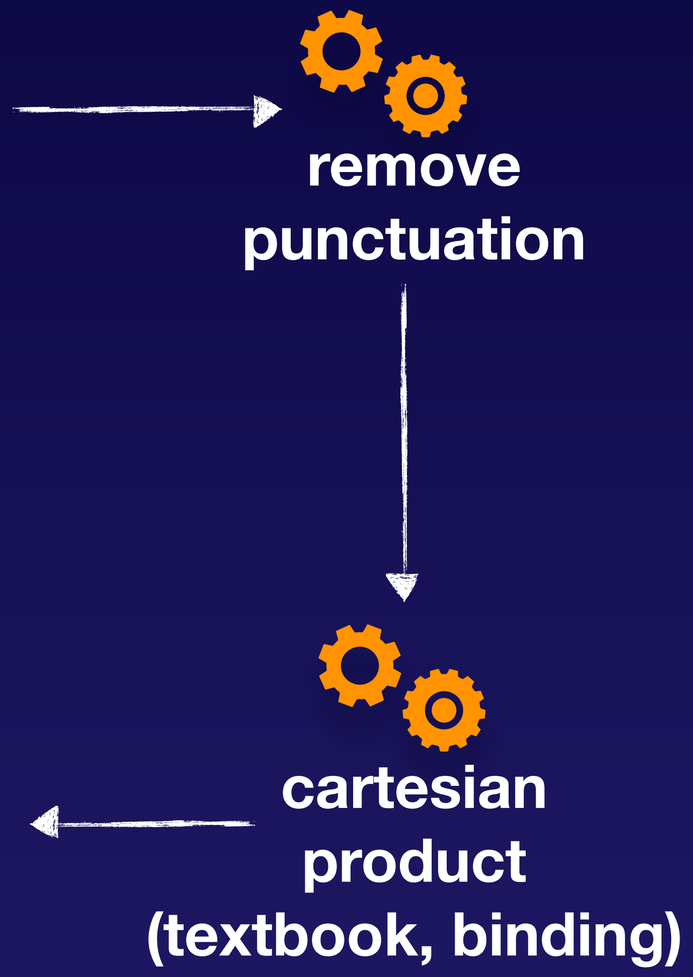
We are multiplying a set of words by another set of words to create a new feature…

Cartesian Product Example

ID	textbook	binding
1	Python Data Science Handbook	Softcover
2	Visualization Analysis & Design	Hardcover
3	Machine Learning Algorithms	Softcover

Cartesian Product Example

ID	textbook	binding
1	Python Data Science Handbook	Softcover
2	Visualization Analysis & Design	Hardcover
3	Machine Learning Algorithms	Softcover



ID	cartesian_product
1	{“Python_Softcover”, “Data_Softcover”, “Science_Softcover”, “Handbook_Softcover”}
2	{“Visualization_Hardcover”, “Analysis_Hardcover”, “Design_Hardcover”}
3	{“Machine_Softcover”, “Learning_Softcover”, Algorithms_Softcover”}

A CLOUD GURU

Feature Engineering Dates

Translating dates into useful information.

A CLOUD GURU

Feature Engineering Dates

Translating dates into useful information.

Feature Engineering Dates

DATE FORMATS

- 2013-02-08T09
- 6 Mar 17 21:22 UT
- Mon 06 Mar 2017 21:22:23 z
- 09/28/2019

EXTRACTED INFORMATION

- Was it a weekend? Was it a week day?
- Was the date the end of a quarter?
- What was the season?
- Was the day a holiday?
- Was it during business hours?
- Was the World Cup taking place on this date?

Feature Engineering Dates

Date
2015-06-17
2015-04-24
2015-02-12
2015-12-16
2015-03-14

date feature engineering

is_weekend	day_of_week	month	year
0	2	6	2015
0	4	4	2015
0	3	2	2015
0	2	12	2015
1	5	3	2015

Text Feature Engineering Summary

Technique	Function
N-Gram	Splits text by whitespace with window size *n*
Orthogonal Sparse Bigram (OSB)	Splits text by keeping first word and uses delimiter with remaining whitespaces between second word with window size *n*
Term Frequency - Inverse Document Frequency (tf-idf)	Helps us determine how important words are within multiple documents
Removing Punctuation	Removes punctuation from text
Lowercase	Lowercase all the words in the text
Cartesian Product	Combines words together to create new feature
Feature Engineering Dates	Extracts information from dates and creates new features