

**Improving the space efficiency of the MS2LDA.org web application**

Radu Petrescu

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at the University of Glasgow

September 1st, 2019

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: Radu Petrescu Signature: *Radu Petrescu*

Acknowledgements

The author would like to thank

the project supervisor, **Dr Simon Rogers**

and **Dr Joe Wandy**, who served as secondary supervisor,

for their support, guidance and patience throughout this project.

Abstract

Mass spectrometry is an analytical technique used in numerous areas such as pharmaceutical drug development, medical diagnostics and metabolomics to name a few. The mass spectrometry workflow can be simplified to represent an input/output system: an input substance is analysed by a mass spectrometer device to derive raw data as output. This data is subsequently processed and analysed *in silico* in order to yield valuable information to the user.

The aim of this project is to improve the space efficiency of the MS2LDA.org web application. The latter deals with *in silico* processing of mass spectrometry data in the field of metabolomics using the text mining algorithm known as Latent Dirichlet Allocation. Metabolomics is the field representing the study of metabolomes, small molecular end-products.

Starting from the working copy of the above-mentioned web application’s code, the project first designs a prototype that calculates variational inference parameters without the need for their permanent storage on the hard disk. Using the knowledge gained thereof, the existing code is then refactored to make the application more space efficient.

[to add a comment about conclusion, limitation, discussion and feasibility]

Contents

Chapter 1 Introduction 1

1.1 Context 1

1.2 Aim 1

1.3 Structure 1

Chapter 2 Analysis & Requirements 2

2.1 Statement of Problem & Motivation 2

2.2 Background 3

2.2.1 Latent Dirichlet Allocation 3

2.2.2 LDA in Mass Spectrometry & Metabolomics 6

2.3 Gathering Requirements & Feature List 7

2.4 Surveyed literature and tools 7

Chapter 3 Design & Implementation 8

3.1 Overview & Implementation Plan 8

3.2 Stage I: Setup & Preparation 8

3.3 Stage II: Prototype Development 9

3.4 Stage III: Refactoring Existing Code 13

Chapter 4 Testing & Evaluation 14

4.1 Testing & Evaluation Strategy 14

4.2 Quality Assurance 14

4.2.1 Stage I Testing: Setup & Preparation 14

4.2.2 Stage II Testing: Prototype Development 15

4.2.3 Stage III Testing <to be written> 17

4.3 Evaluation results 17

Chapter 5 Conclusion 18

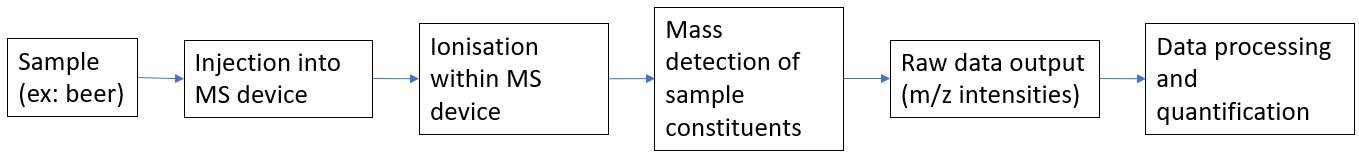
Chapter 6 Bibliography 19

Appendix A More bib, tests, queries, extras 1

# Introduction

## Context

Mass spectrometry (hereby “MS”) is a technique for measuring the mass-to-charge (“m/z”) ratio of the ionised particles of a chosen sample.[[1]](#endnote-1) The generated spectra signals are used to identify the presence and characteristics of elements and molecules in the sample. [[2]](#endnote-2) A simple diagram of this workflow is presented below:



As an analytical technique, MS is essential in numerous areas such as pharmaceutical drug development [[[3]](#endnote-3)], metabolomics [[[4]](#endnote-4)] and environmental testing [[[5]](#endnote-5)] to name a few. Taking into consideration the diagram above, this project is only concerned with the final stage, namely the data processing and quantification of the raw output from the MS device (hereby referred to as “*in silico* processing”). *In silico* processing is currently done through the web application MS2LDA.org that maps the masses of individually identified fragmentation features (sample constituents) into motifs.[[6]](#endnote-6) Motifs, concisely, are visual representations showing a pattern of related constituents, as shown in the following diagram:[[7]](#endnote-7)



The application utilises Latent Dirichlet Allocation (hereby “LDA”), a text mining algorithm that will be further expanded upon in the following sections.[[8]](#endnote-8) MS2LDA.org has a corresponding connected database named MotifDB, which stores motifs and corresponding data.[[9]](#endnote-9)

## Aim

Although the functionality of MS2LDA.org is not contested, the web application is heavily resource intensive in terms of the hard disk memory required.

**The aim of this project is to modify and enhance the MS2LDA.org web application in order to increase its space efficiency and reduce the amount of non-volatile memory consumed on its host server.**

## Structure

The report is structured into the following main sections:

1. Analysis & Requirements;

2. Design & Implementation;

3. Testing & Evaluation;

# Analysis & Requirements

## Statement of Problem & Motivation

This report will hereby refer to the owners of the MS2LDA.org application as “the Client”.

The Client currently has a functionally working version of the web application available online at ms2lda.org. The application runs on a server with limited space resources that are increasingly running scarce due to the significant amount of data produced by the application. This is the Client’s problem.

The solution to this problem consists in refactoring the system so that it consumes less space resources without impacting the functionality of the web application. The functionality of the application implies both that its behaviour is unchanged and that its computational speed is not significantly reduced.

As it will be further explained in the following sections of the report, much of the data stored by the web application on the server can be computed at run time and therefore needs no long-term storage on the server. This idea will be leveraged in producing the desired solution for the Client.

MS2LDA.org deals with large mass spectrometry data sets from experiments in the field of metabolomics.[[10]](#endnote-10) This field is concerned with the study of metabolomes, small molecular end-products.[[11]](#endnote-11) There is a growing interest in combining metabolomics data sets from multiple sources in order to yield comprehensive scientific insights and possibly new discoveries.[[12]](#endnote-12) The main bottlenecks to such developments are the general lack of integrative analytical tools and software[[13]](#endnote-13) as well as the lack of labelled data.[[14]](#endnote-14) In particular, the difficulty of obtaining labelled data is significantly influenced by the fact that data sets can take a great amount of hard disk space due to the multitude of data points in the samples (samples can easily reach millions of data points).[[15]](#endnote-15)

The main motivation of this project is therefore not only concerned with reducing the space burden for a specific server and satisfying the Client, but with providing insights into ways that the space efficiency of data sets and metabolomics tools/software can benefit the scientific community at large and improve the likelihood of advancements in metabolomics, related fields and life sciences in general.

One of the motivations of MS2LDA.org upon its creation was to tackle the abovementioned bottlenecks by providing a method for molecular fragment data set decomposition in the field of metabolomics.[[16]](#endnote-16) The web application is not merely a prototype, but a real functioning system where interested parties from the scientific community can upload their own mass spectrometry data, run analyses and view relevant visualisations of the results. Enhancing the web application therefore is intended to provide a focused benefit not only to the direct Client, but also to its vast user group in the scientific and medical community.

The following section conveys the relevant theoretical background to the project.

## Background

### Latent Dirichlet Allocation

MS2LDA.org uses the text mining algorithm known as LDA adapted for the fields of mass spectrometry and metabolomics.[[17]](#endnote-17) Understanding LDA is crucial for the scope of this project since comprehending the underlying algorithm allows for a more space efficient computation of the relevant data.

LDA is a generative probabilistic model that is commonly used, among other areas, for topic modelling in text corpora.[[18]](#endnote-18) A text corpus is defined as a large and unstructured collection of text documents containing discrete text data. Each document, the sub-constituent of the corpus, is a collection of words. The word can be defined as the atomic discrete unit of text data. The set of all the words in the corpus is referred to as the vocabulary. LDA assumes that documents are distributions over topics, and that topics are distributions over words. A topic is probability (multinomial) distribution over words. Put differently, words have a certain probability of being part of a specific topic.[[19]](#endnote-19) This is best illustrated through an example: in a corpus, words such as “football” and “championship” are more likely to belong to the topic “sports” as opposed to the words “cat” and “dog”, the latter having a high probability of being part of the “animals” topic. It must be noted that topics are considered latent/hidden in the LDA model and therefore specific labels such as “sports” are for illustration purposes only.

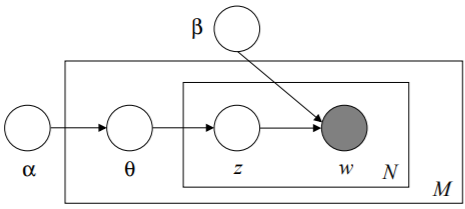
The LDA model assumes that the number of topics is known, fixed and finite. This helps us determine the hyperparameter known as β (Beta). A hyperparameter is simply a parameter applicable to the entire corpus, while Beta is the topic distribution over words and is represented as a matrix where each row is a topic, each column is a unique word in the vocabulary and their intersection cell is the respective probability. Each row must be normalized, summing up to the value of 1. The following figure represents an example Beta matrix for a hypothetical corpus made up of three topics and three words (the grey shaded area is the matrix):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Word 1** | **Word 2** | **Word 3** |
| **Topic 1** | 0.1 | 0.2 | 0.7 |
| **Topic 2** | 0.2 | 0.2 | 0.6 |
| **Topic 3** | 0.4 | 0.4 | 0.2 |

The second hyperparameter relevant in the Dirichlet context is referred to as the α (Alpha) parameter. The Alpha parameter is sometimes referred to as the concentration parameter as it represents the probability of the topics appearing in the documents of the corpus. Note that Alpha is not a probability *per se* since it can have values larger than 1.[[20]](#endnote-20)[[21]](#endnote-21) Alpha is therefore a vector (a 1-dimensional matrix) with each concentration value corresponding to a specific topic. The following figure illustrates an Alpha vector for a 3-topic corpus (the grey shaded area is the vector):

|  |  |
| --- | --- |
|  | **Alpha (concentration) value** |
| **Topic 1** | 0.5 |
| **Topic 2** | 0.4 |
| **Topic 3** | 0.1 |

Visualising the interconnection between the elements described in this sub-section can be best described using a plate diagram:[[22]](#endnote-22)



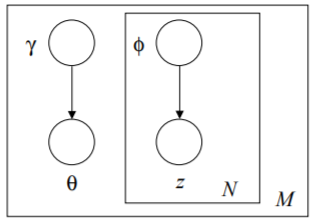
In the above diagram α and β are the abovementioned hyperparameters standing for topic concentration values and topic over word probability distribution respectively. The M plate represents a single specific document. The N plate represents a single specific word in a specific document. The latter plate contains two elements. The first element, w, represents the occurrence of a word. As it can be observed from the arrow directions, each such word occurrence corresponds to a specific topic z (the second element) and to a specific word/topic probability from the Beta matrix. It is known that Alpha is a concentration parameter for all the topics in the corpus, but in order to get the topic probabilities for a specific document, a θ (Theta) parameter is created. The latter is simply a vector of the probabilities of all the topics occurring in a specific document. Assuming a corpus of three topics, specific document M’s Theta vector is illustrated through the gray-shaded area in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Topic 1** | **Topic 2** | **Topic 3** |
| **Document M** | 0.3 | 0.3 | 0.4 |

Note that the Theta vector is normalized as well, meaning that the probabilities for each row must sum to 1. To derive the entire document to topic probability distribution, the respective Theta vectors of all documents in the corpus can be concatenated in order to produce a two-dimensional matrix with documents as rows and topics as columns.

The plate diagram above conveys another important aspect that needs to be taken into consideration. The w parameter is the only one that is grayed since it is the only element that is observed.[[23]](#endnote-23) LDA is called “latent” because most of its parameters are hidden and therefore must be inferred. Inference, in this scenario, can be defined as the means to derive conclusions about the unknown parameters of a sample/corpus.[[24]](#endnote-24) There are various methods for inference regarding LDA such as Markov-chain Monte Carlo methods (example: Collapsed Gibb’s sampling), Online (Stochastic) methods or Variational Inference methods.[[25]](#endnote-25) This report will only proceed to go into detail concerning variational inference since this is also used by the MS2LDA.org web application.

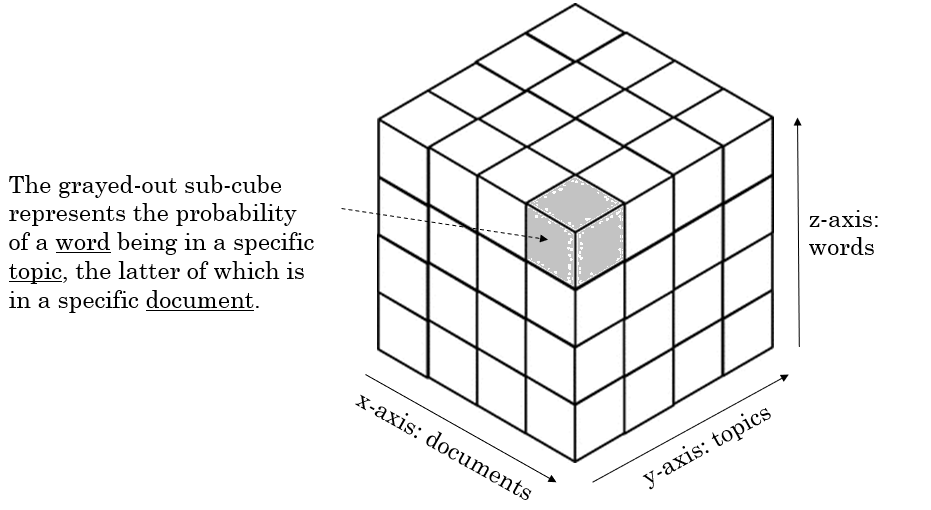
Variational inference works by approximating the target distribution of hidden parameters in LDA by minimising the difference between the target distribution and simpler variational distributions. In order to apply variational inference, two variational parameters are needed: γ (Gamma) and φ (Phi). Their relationship to the previous LDA diagram is conveyed in the following secondary plate diagram:



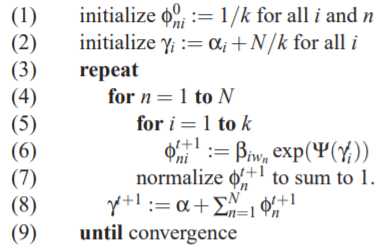
Just like in the first plate diagram above, M is a single document, N is a single word in M, z is the topic corresponding to N and Theta is the distribution of topics per document.

Gamma is very similar to Theta in that it is a vector illustrating the topic incidence for a particular document. The difference lies in the fact that Theta is a normalised version of Gamma. Theta is also sampled from Gamma. If all the Gamma vectors for all documents are concatenated, a so-called two-dimensional Gamma matrix is created with each row representing a document and each column representing a topic for the given corpus.

Phi is the collection of probabilities of words appearing in topics for specific documents. In other words, phi can be mapped into a three-dimensional matrix resembling a cube composed of substituent smaller cubes, each of the latter representing a single probabilistic value. The x-axis represents documents, the y-axis represents topics and the z-axis represents specific words. This abstraction can be visualized as in the following “cube-of-cubes” diagram:[[26]](#endnote-26)



Variational inference follows a two-step procedure. The first step is called the e-step and involves estimating the two variational parameters Gamma and Phi. The second step is referred to as the m-step and is concerned with re-estimating the hyper-parameters Alpha and Beta given the data in the e-step.[[27]](#endnote-27) This process can be summarised through the following iterative process:[[28]](#endnote-28)



The first two steps initialise the variational parameters Phi and Gamma respectively. Steps 3 to 9 illustrate a nested for loop that repeats until convergence occurs. Convergence occurs when the difference between the values of Beta for two iterations is small enough to be considered insignificant in hindering appropriate inferential approximation. Beta is reset through each of its respective iterations and so is Alpha for its respective iteration. The above algorithm is implemented in the web application through the methods: *init\_vb* (for initialisation), *e-step* (for the above loop, though with slight modifications to perform calculations in log space) and *vb\_step* (the actual resetting/updating of the hyperparameters Alpha and Beta). Ψ above stands for the Psi / Polygamma function.[[29]](#endnote-29)

In order to achieve the objectives of the project, the variational inference steps provide a powerful insight: only the corpus original data and the initial Alpha and Beta parameters are needed to be stored on the hard-disk / in a database. The rest of the parameters can be initialised, computed or updated without having a need for their permanent storage. In other words, the variational parameters Phi and Gamma, as well as the latter’s normalised version (Theta) would only occupy space redundantly if stored in a database. This crucial insight will be further expanded upon in the Design & Implementation section of the report below in order to progress towards achieving the objective.

The next subsection focuses on how mass spectrometry and metabolomics scientific jargon can be mapped to the LDA terminology discussed in this subsection.

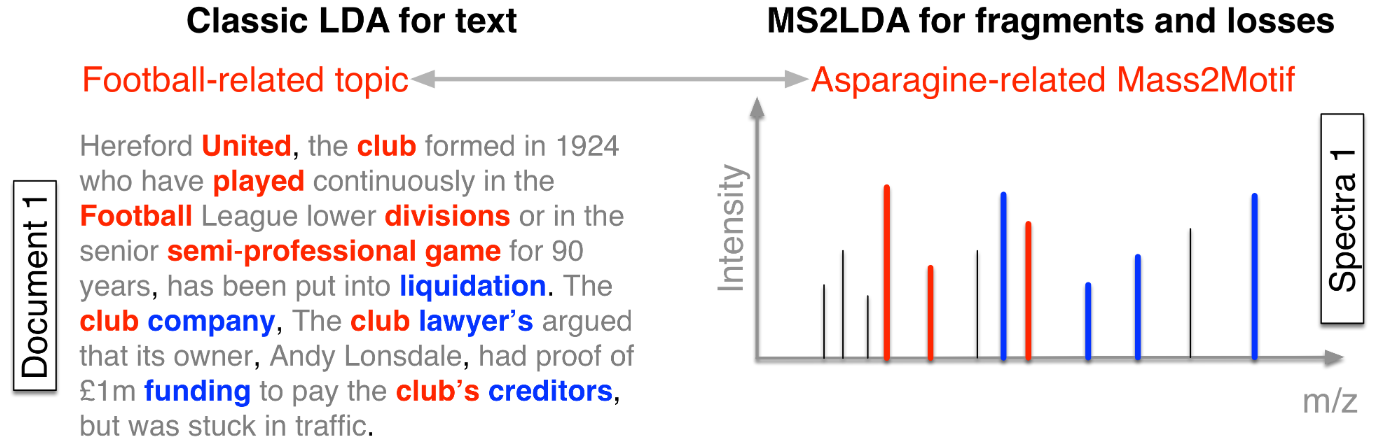
### LDA in Mass Spectrometry & Metabolomics

A brief and simplified introduction to mass spectrometry and metabolomics has been provided in the Introductory sections of this report. As the objective of this report is purely focused on the non-functional requirement of increasing space efficiency, a deep understanding of the actual functional aspects of mass spectrometry and metabolomics is not necessary. However, it is important to be able to map scientific jargon to the LDA terminology presented in the previous subsection.

It can be noticed above that Alpha, Beta, Gamma, Theta and Phi can be expressed through the concepts of corpus, words, topics and documents. The following table provides a mapping to these four key concepts:

|  |  |
| --- | --- |
| **Mass Spectrometry Concept** | **LDA Concept / Term** |
| Feature (or Fragments/Losses) | Word |
| Motif (or Mass2Motif) | Topic |
| Spectrum (or just Document) | Document |
| Experiment / Original data | Corpus |

Visually, the mapping above can also be observed in the following diagram[[30]](#endnote-30):



## Gathering Requirements & Feature List

There are many traditional techniques for gathering requirements such as user interviews, questionnaires and workshops.[[31]](#endnote-31) This project uses a pragmatic tri-partite approach that is described below. There are two important aspects of this report: the software (MS2LDA.org) already exists in a functional state and the objective of the report does not give rise to any functional requirements. Taking these into consideration, the requirements gathering process is composed of:

1. Initial communication with the Client in order to establish the objective and scope of the project (minutes of the meetings are kept in paper form);
2. Periodical (weekly) meetings with the Client in order to clarify any arising issues or in order to perform acceptance testing (minutes of the meeting are kept);
3. Constant contact and communication with the client on an as-required basis using collaboration tools such as Slack for any immediate questions or bottleneck resolutions;

The above requirement gathering process is the most pragmatic and suitable in this context since the Client is in the best position to judge the non-functional successes of the project as well as the success of the preservation of existing functional aspects.

In terms of the actual requirements, there is only a single mandatory must have non-functional feature:

**The web application must use less hard disk space.**

This requirement, however, can be quite vast and through iterations in providing suitable solutions to it one may find that further “should-have” and “could have” requirements result from such iterations. For example, a space-time trade-off is quite common in software and algorithms.[[32]](#endnote-32) As such, given the context, the project takes a pre-emptive approach and is expecting other potential requirements to arise such as:

|  |  |
| --- | --- |
| **Priority** | **Potential Secondary Non-Functional Requirements** |
| Should Have | The web application must not have a significant or noticeable drop in time efficiency. |
| Could Have | The web application uses a caching system in order to support an added value computational time efficiency. |

As it can be observed, potential requirements are adaptive and change with the context. They are also heavily dependent and conditional on the main must-have feature and its implementation. Features have therefore developed with the project, with the main feature in mind. The final version of the ordered list of features, their justification and aims/objectives are presented in appendix 1. **<TBCOMPLETED -include Moscow statements and user stories/constr>**

## Surveyed literature and tools

As MS2LDA.org is a custom-made system and since the objective is to enhance it, competing or similar products have not been heavily scrutinized. A complete list of surveyed literature and software tools is presented in the bibliography.

# Design & Implementation

## Overview & Implementation Plan

For the scope of this project, MS2LDA.org is not a project that is designed and developed *ab initio*. The web application is already available and functional, having its own niche user group. Given this important aspect, the project represents merely an enhancement that does not introduce any regression bugs or performance concerns.[[33]](#endnote-33) As the previous section illustrates, there is only one main non-functional requirement: reduce hard disk space. The end product remains unchanged from both the user’s perspective and that of the system architect.

Taking into account the LDA theory presented in the Background section above and upon confirmation during one of the initial meetings with the Client, the variational inference parameters need not be stored in the long term on the hard disk. In other words, the parameters Phi and Gamma (and the latter’s normalised version, Theta) are merely used for updating the hyperparameters Alpha and Beta. Once the updates are completed, the former three parameters can be deleted. In practical terms, this translates our non-functional requirement into refactoring the existing code so that the three variational inference parameters are no longer stored in the database but only used real-time as the program runs.

In order to achieve this, three main categories of resources are obtained from the Client as inputs:

1. A working copy of the code for the current instance of the MS2LDA.org web application, available on GitHub at <https://github.com/sdrogers/ms2ldaviz>;
2. A PostgreSQL database dump containing at least one mass spectrometry experiment’s data that has been verified by the Client, allowing for testing to be performed between original and calculated outputs;
3. Weekly meetings with the Client in order to set intermediary short term objectives and test the prototype/product incrementally from the user’s perspective (constant communication was also maintained via Slack with the Client’s team)[[34]](#endnote-34);

Given these inputs, the project can be divided into three stages:

1. Stage I: Setup & Preparation;
2. Stage II: Prototype Development;
3. Stage III: Refactoring Existing Code;

This section will now proceed to consider each stage of the tripartite implementation plan.

## Stage I: Setup & Preparation

The objective of this stage is to get a local instance of the web application to run using the database dump provided by the Client and to find a suitable medium to develop the output of stage II, the prototype.

In order to achieve the above objective, a bottom-up approach was taken: the database layer was configured first. The database backup was received in PostgreSQL’s .dump[[35]](#endnote-35) format and therefore had to be manually restored using the following terminal command:

pg\_restore --if-exists --verbose --clean --no-acl --no-owner -h localhost -p 5433 -U postgres -d postgres ms2ldaviz\_db.dump

DBeaver, an SQL client and database administration tool was used for checking the correctitude of both the structure and data corresponding to the database.

With the database layer set up, the Django/middleware layer was tackled next. The main GitHub repository was forked in order to reduce any risk of influencing the original code inappropriately. The repository was pulled locally and a virtual environment was created using Python’s pipenv. The underlying Python module requirements were then downloaded. Some of the requirements were out of date and had to be imported manually using Python’s ‘pip install’ commands. The IDE used for the project was PyCharm. Once the settings.py file was correctly configured in the project folder locally, the connection to the database was established and the web application worked flawlessly.

Upon discussion with the Client, the decision was made that the best environment for prototyping would be a Jupyter Notebook. One such Notebook file was created in a project folder with existing notebooks and was appropriately configured to the MS2LDA.org code. This marked the start of the stage II, the Prototype Development phase.

## Stage II: Prototype Development

The objective of this stage is to create a prototype that can be used to calculate the variational parameters in real time without the need for hard disk storage or refactoring in the original code. Upon verification of the viability of the prototype by the Client, the project can proceed to stage III, which represents the refactoring of the original MS2LDA.org code.

The first step involved discussing and choosing with the client the main mass spectrometry experiment to be scrutinised. The choice made was massbank\_binned\_005 (having experiment ID 190). This was known to be a 500 LDA topic experiment with a specific number of words and documents that could be easily tested due to its assumed data correctitude. With the experiment chosen, the required input data was extracted from the database. The input model in our case consists of the corpus data (the actual words, documents and topics for the chosen experiment), the Alpha hyperparameter data and the Beta hyperparameter data. The data was retrieved from Django’s models and tested against direct SQL queries with the one of the Client’s team member’s supervision. Each LDA element (word, document or topic) ID was mapped to a specific prototype index to identify and set the position of that specific element for the entire experiment. An example of code for topic retrieval and mapping is given below:

mi = Mass2Motif.objects.filter(experiment=experiment)

unique\_topics = {}

index=0

for m in mi:

unique\_topics.update({m.id:index})

index+=1

A similar approach was taken for words and documents as well. Once this was done, the three LDA elements were combined into what can be referred to as the dictionary representation of the corpus, which is a nested Python dictionary of the form: {document:{word:wordcount}}.

The Alpha hyperparameter was determined by mapping the Alpha values from the database to corresponding topic positional indexes in order to create a NumPy array, the latter being equivalent to a vector. The code fragment for the vector value assignment is as follows:

alpha\_vec = np.zeros(n\_motif)

for pos,val in alphas.items():

alpha\_vec[pos] = val

The Beta hyperparameter represents a matrix corresponding to a mapping of words to topics. In order to derive it, an empty NumPy two-dimensional array is created with length equal to unique topics and width equal to unique words. This is similar to a 2D matrix. The program then iterates through all Beta probability values received from the database and assigns them to the corresponding array position. Afterwards, Beta is normalised through the following algorithm:

i = 0

while i<K: #...for each row of the Beta matrix

row = pivot\_table\_normalised[i, :]

adjusted\_row = row + SMALL\_NUMBER

normalised\_row = adjusted\_row / np.sum(adjusted\_row)

np.sum(normalised\_row)

pivot\_table\_normalised[i, :] = normalised\_row

i+=1

Please note that the above algorithm is used whenever normalisation is necessary. With the input data now available for calculation of the variational parameters, the prototype then obtains the original variational parameter values from the database. The original values are important in that they allow for comparison with the calculated values that will be the output of the prototype. If the values approximately coincide, the prototype is deemed viable and the project is able to proceed to the third stage, namely the refactoring of the original code.

The Gamma variational parameter is extracted from the models in its normalised Theta form. The original Phi values are extracted from the models by joining the Feature Instance models to the Feature Mass2Motif models. This gives us the original variational parameters in list form, allowing for easy comparison to the calculated values.

The calculated variational parameters Gamma and Phi must first be initialised. Calculated Phi is initialised as a dictionary of the form {document:{word:topic\_vector}} with 0 values populated by NumPy. Each calculated Gamma vector is initialised by adding the Alpha vector received above to *1.0\*(doc\_total/K)*, the latter being the float value of the division of a document word count by the number of topics. Once the initialisation is completed, the e-step is then performed.

The e-step is a nested for loop that iterates over each document and subsequently each word so that it updates the calculated variational parameters until the point where Gamma converges to 0. The code for both the initialisation and the e-step were reused from the original MS2LDA.org repository and the following code snippet illustrates how the code was transcribed and adapted for the purpose of this project’s prototype:[[36]](#endnote-36)

for i in range(iterations):

prev\_gamma = np.copy(current\_gamma)

for doc in corpus:

d = int(doc)

doc\_dict = corpus[doc]

temp\_gamma = np.zeros(K) + alpha\_vector

for word in doc\_dict:

w = int(word)

log\_phi\_matrix = np.log(beta\_matrix[:,w]) + psi(gamma\_matrix[d,:]).T

log\_phi\_matrix = np.exp(log\_phi\_matrix - log\_phi\_matrix.max())

phi\_matrix[d][w] = log\_phi\_matrix/log\_phi\_matrix.sum()

temp\_gamma += phi\_matrix[d][w]\*corpus[doc][word]

temp\_beta[:,w] += phi\_matrix[d][w] \* corpus[doc][word]

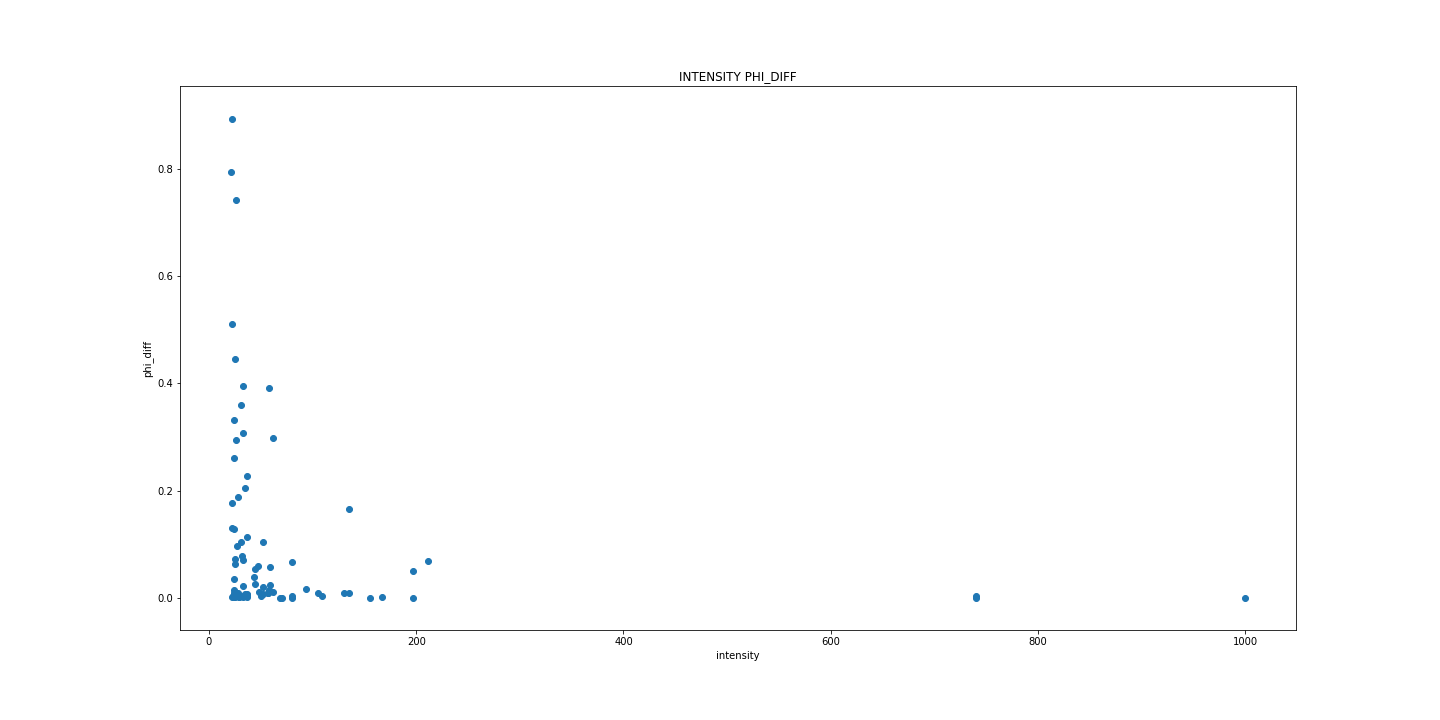
gamma\_matrix[d,:] = temp\_gamma

pos = np.where(gamma\_matrix[d,:]<SMALL\_NUMBER)[0]

gamma\_matrix[d,pos] = SMALL\_NUMBER

current\_gamma = np.copy(gamma\_matrix)

The variational parameters are then compared. It was noted that using the whole corpus would take too much time for the prototype and therefore this was implemented for individual documents only. Comparison was performed for multiple documents individually in order to test the hypotheses. The initial prototype comparison yielded great differences in both Phi values and normalised Gamma values. The Phi value differences were found to be high only for immaterial words, namely those words that have low intensities. Trend wise, the differences also decreased for higher word intensities. This was proven by plotting word intensities (x-axis) versus Phi difference (y-axis) on a scatterplot. The following diagram is an example for the document with database ID 269323:



Even though the Phi differences were proven and agreed with the Client as being correct and acceptable, normalised Gamma differences still remained large and unexplained. Each material topic in the normalised Gamma data was found to have a high percentage of low intensity/immaterial words using the following code:

topics\_tested = np.array(output\_data\_theta)[:,1]

temp =[]

for t in topics\_tested:

temp.append(int(t))

topics\_tested = temp

print("topic", "material", "immaterial")

for topic in topics\_tested:

topic\_tested = topic

count\_material = 0

count\_immaterial = 0

for line in phi\_original:

if line[2] == topic\_tested:

if line[1] not in material\_words\_list:

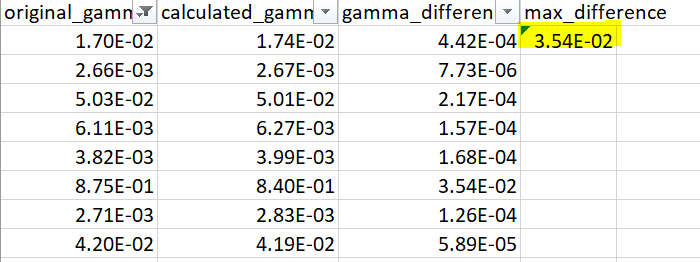
count\_immaterial += 1

else:

count\_material += 1

print(topic\_tested, count\_material, count\_immaterial)

For example, for the same document with database ID 269323, it’s most significant topic (90% probability) for the tested document had over 90% of its words below a materiality level of 2% of the maximum word intensity value (1000). This led to the hypothesis that the gamma differences could be explained by thresholding. Thresholding is performed by the web application in order to eliminate low intensity words that would otherwise qualify as noise. In order to test this hypothesis, two input files with experiment data were received from the client. Both files concerned the same experiment but had different thresholding levels (high and low). A second version of the prototype was created in order to parse the data from the input files. For the higher thresholding input file, the same divergences in normalised Gamma occurred. For the lower thresholding input file, however, normalised Gamma differences were very low. For example, for the document with database ID 270414, the maximum Gamma difference was just above 3.5%, as shown in the following snapshot of the Gamma difference output file:



The differences were verified for several documents and it was agreed with the Client that the results are correct. With the prototype calculations now viable, the project proceeded to the next stage, the refactoring of the existing MS2LDA.org code.

For a more in depth view of the prototype code, please visit <Final GitHub Link>.

## Stage III: Refactoring Existing Code

* LEFT OFF HERE FOR IMPLEMENTATION PART
* Refactor a page, then do entire refactoring.
* Objective…
* How it was done…
* Details, outcome, explanations, etc. (similar to above)

# Testing & Evaluation

## Testing & Evaluation Strategy

The software testing strategy closely follows the three phases of the implementation presented in Chapter 3 above. As such, testing was performed in tandem with setup and development. The specific tests for each implementation stage section will be presented in the Quality Assurance section below.

The testing and evaluation approach insisted on two principles:

1. Engage the client as much as possible in the testing process;
2. Adopt a modified test-driven-development (TDD) approach; [[37]](#endnote-37)

For the second principle employed, TDD traditionally stands for developing (mostly automated) tests first and then ensuring that the code satisfies such tests. In our case, a similar paradigm was employed – we first wrote down the objective of the specific design/implementation stage and then tested it accordingly both through code and through the user testing (the Client’s testing input).

This report refers to this strategy as a Teleological Test Driven Development, the latter being teleological since a test can only be satisfied once the objective of its respective stage has been met. The strategy assumes that there are no legacy bugs or initial issues in the code provided by the Client.

## Quality Assurance

### Stage I Testing: Setup & Preparation

This stage involved mostly manual testing and user acceptance testing since at this point there was little familiarity with the web application and guidance was needed in order to provide a clear understanding of what the correct setup is.

**Objective 1: The database dump must be complete and correct.**

The structure and data of the database dump was checked using the DBeaver tool together with one member of the Client’s team. Particularly, the data of the prototype experiment massbank\_binned\_005 was checked using SQL queries in the relevant tables of the database. With the acceptance of the Client, the database layer was assumed to be in a functional and correct state.

**Objective 2: The local web application instance setup must be correct.**

After the resolution of the underlying Python requirements and database connectivity issues in the settings.py file, the local instance of the web application was checked versus the remote production version of the database to spot any significant differences, especially for experiment 190. No differences were found, a member of the Client’s team also confirmed the correctitude and the project was assumed ready for progression to the next stage.

### Stage II Testing: Prototype Development

Although this section could have used more complex testing such as Python’s PyTest or UnitTest frameworks, the minimal marginal benefit provided by automation to the project and the prototype would not have justified the time investment. As such, most of this section was tested using more technical manual tests involving Python code in Jupyter Notebook and SQL querying.

**Objective 1: All corpus elements (topics, documents, words) must be correctly imported in the prototype.**

Knowing that the database data and structure had already been tested above, this objective was tested by first comparing the length of the data structures containing each element to the count of the corresponding elements in the database. These must coincide. For example, in the case of corpus topics, len(unique\_topics) and the following SQL query should yield the same result, namely 500:

select count(\*) from basicviz\_mass2motif where experiment\_id=190

The same was done for words and documents. Data items were also compared for coincidence using .csv exports from NumPy and Microsoft Excel filters. The results coincided and therefore this objective was assumed to be passed.

**Objective 2: Alpha and Beta NumPy Arrays have correct positioning.**

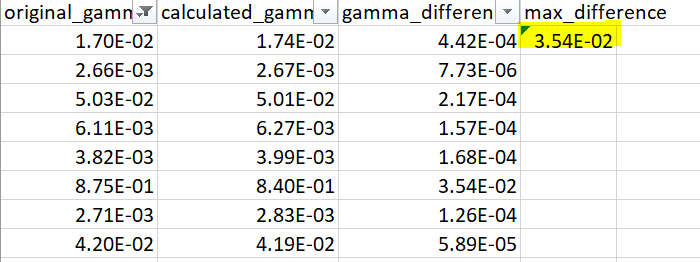
* [partially complete section]
* Checked position of a few random topics for alpha using csv export.
* Checked position for a few random specific [topic,row] pairs in beta such as [12, 12790, 0.0333402023123277] using csv export from numpy.
* Correct results obtained.

**Objective 3: Original Phi and Gamma correspond to database queries.**

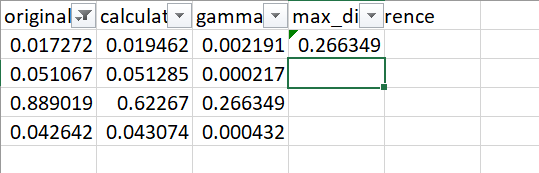
* [partially complete section]
* For more complex Django model imports, this was checked against specific queries in the database.
* Show query versus model results (selectively for phi – some random specific rows).
* Correct results obtained. (more a form of double checking)

**Objective 4: Variational Inference parameters correspond (original versus calculated).**

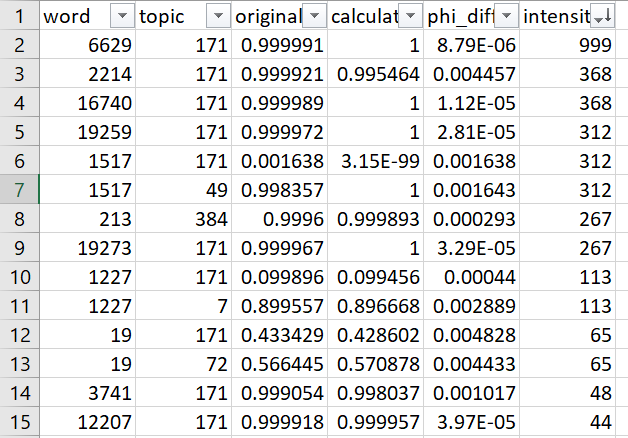
* [partially completed section]
* For gamma comparison we have the gamma\_compare.csv output files. For low thresholding the results are as follows:



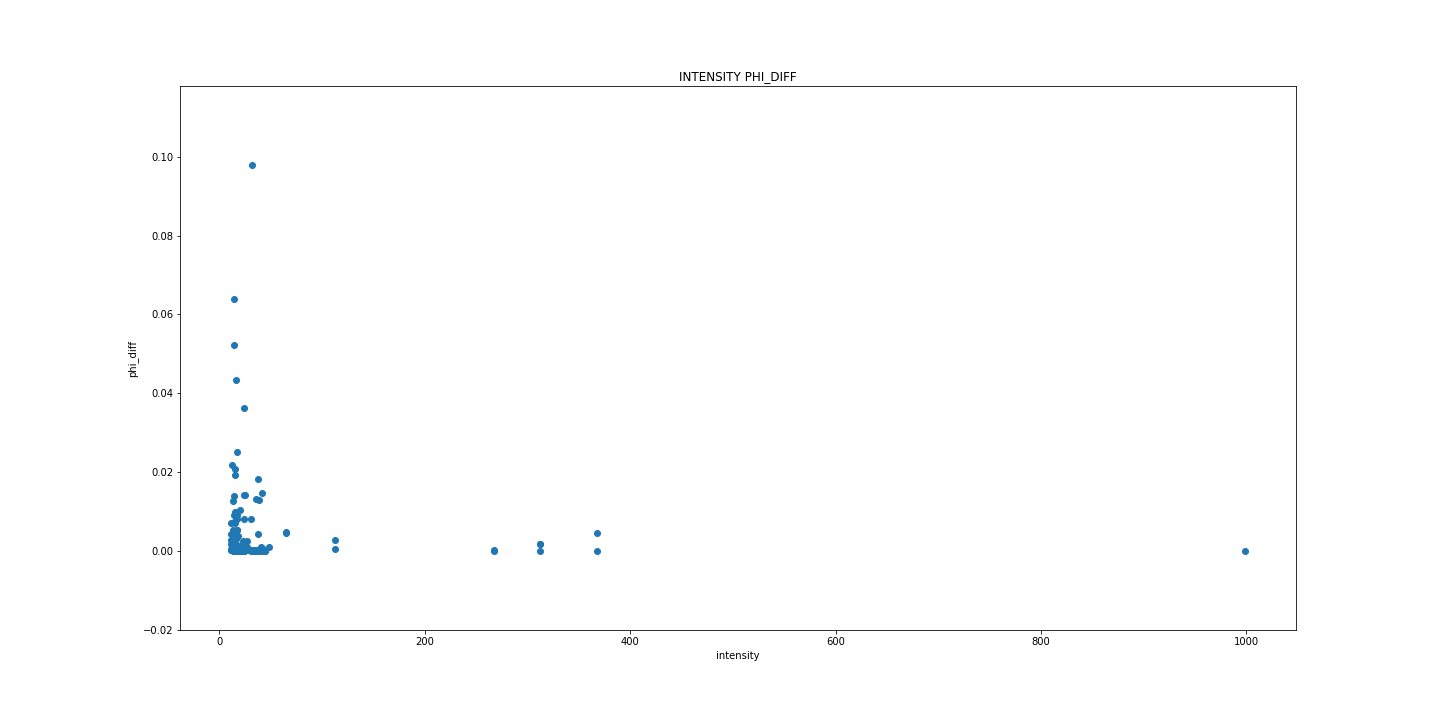
* For gamma the results did not correspond initially and it was assumed this was to thresholding. Version 2 of prototype was created with 2 high/low thresholding input files being tested against each other. For high thresholding, we have results below, for low thresholding we have results above:



* For phi we may have complete differences (phi values are up to 1, so some differences might almost converge to one and be of the form 0.999…) but this was proven to be only for low intensity words. For high intensity words, phi differences were very small as shown by both csv exports(sorted from highest intensity to lowest)…:



…and the scatterplots:



* Prototype works for one document at a type. Multiple documents were tested. Client was consulted and happy.

### Stage III Testing <to be written>

<to be written upon completion>

## Evaluation results

* Did it meet our objectives/requirements?
* How did it meet our objectives/requirements?
* Mention the good and the bad. (reflection)
* Specify areas of improvement.
* Specify evaluation criteria (usability, space reduced, computation time, etc…)
* <to be completed at end of coding session>

CONTINUE FROM HERE

# Conclusion

Python 3 might be a safer option – definitely talk about this.

2-3 pages

Conclusions should state the achievements, and a reflection on achievements (what wasn't achieved/ why and what more could have been done or done differently in hindsight - pick one or two issues to discuss in \*depth\* rather than trying to be comprehensive. should include future work as well.

# Bibliography

* https://onlinelibrary.wiley.com/doi/abs/10.1002/0471238961.1301191913151518.a01.pub2
* https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-S7-S9
* https://www.future-science.com/doi/full/10.4155/bio.11.232
* https://onlinelibrary.wiley.com/doi/abs/10.1002/mas.20108
* https://www.sciencedirect.com/science/article/pii/S0021967304021648
* https://github.com/sdrogers/ms2ldaviz ; ms2lda.org, use joe’s reference!!!
* https://www.ncbi.nlm.nih.gov/pubmed/28968802; ms2lda.org/user\_guide
* http://www.jmlr.org/papers/v3/blei03a.html
* <http://eprints.gla.ac.uk/176662/>

<https://github.com/blei-lab/onlineldavb>

<https://www.youtube.com/watch?v=vYIuvSwLsMc>

tango with Django

<https://github.com/ddbourgin/numpy-ml/blob/master/lda/lda_smoothed.py>

<https://www.machinelearningplus.com/python/numpy-tutorial-part1-array-python-examples/>

<https://www.machinelearningplus.com/nlp/gensim-tutorial/>

<https://mimno.infosci.cornell.edu/info6150/readings/HoffmanBleiBach2010b.pdf>

<https://www.youtube.com/watch?v=fCmIceNqVog>

<https://www.youtube.com/watch?v=2pEkWk-LHmU>

<http://times.cs.uiuc.edu/course/598f16/notes/lda-survey.pdf>

<https://github.com/ChangUk/pyGibbsLDA/tree/master/src>

<http://theses.gla.ac.uk/7928/1/2017wandyphd.pdf>

* <https://www.sciencedirect.com/science/article/pii/S1877705811018674>
* <https://www.thoughtvector.io/blog/lda-alpha-and-beta-parameters-the-intuition/>
* <https://datascience.stackexchange.com/questions/199/what-does-the-alpha-and-beta-hyperparameters-contribute-to-in-latent-dirichlet-a>
* <http://www.jmlr.org/papers/volume18/15-595/15-595.pdf>
* <http://times.cs.uiuc.edu/course/598f16/notes/lda-survey.pdf>
* <https://www.statisticshowto.datasciencecentral.com/posterior-distribution-probability/>
* Wandy 2017 -phd thesis joe <http://theses.gla.ac.uk/7928/1/2017wandyphd.pdf>
* Petre Stegaroiu diss level 4
* Joe Frew diss level 4
* <http://metabolomicssociety.org/resources/metabolomics-software>
* <https://medium.com/@lettier/how-does-lda-work-ill-explain-using-emoji-108abf40fa7d>
* <https://onlinelibrary.wiley.com/doi/abs/10.1002/0471238961.1301191913151518.a01.pub2>
* <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5860206/>
* <https://www.pnas.org/content/113/48/13738>
* <http://eprints.gla.ac.uk/147960/>
* <https://www.tandfonline.com/doi/full/10.1080/21670811.2015.1093271>
* <https://www.thoughtvector.io/blog/lda-alpha-and-beta-parameters-the-intuition/>
* <http://videolectures.net/mlss09uk_blei_tm/>
* <https://datascience.stackexchange.com/questions/199/what-does-the-alpha-and-beta-hyperparameters-contribute-to-in-latent-dirichlet-a>
* <https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158>
* <https://www.google.com/search?q=what+are+the+latent+variables+in+lda&rlz=1C1CHBF_en-GBGB841RO842&oq=what+are+the+latent+variables+in+lda&aqs=chrome..69i57j33.6560j1j7&sourceid=chrome&ie=UTF-8>
* <https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd>
* Scott grieg dissertation

− list, in alphabetical order by author and date, all articles that you have consulted. For each article, give full bibliographic details

###### More bib, tests, queries, extras

<Use Heading 6 for the Appendix heading>

1. https://onlinelibrary.wiley.com/doi/abs/10.1002/0471238961.1301191913151518.a01.pub2 [↑](#endnote-ref-1)
2. <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-S7-S9> [↑](#endnote-ref-2)
3. <https://www.future-science.com/doi/full/10.4155/bio.11.232> [↑](#endnote-ref-3)
4. <https://onlinelibrary.wiley.com/doi/abs/10.1002/mas.20108> [↑](#endnote-ref-4)
5. <https://www.sciencedirect.com/science/article/pii/S0021967304021648> [↑](#endnote-ref-5)
6. <https://github.com/sdrogers/ms2ldaviz> ; ms2lda.org [↑](#endnote-ref-6)
7. <https://www.ncbi.nlm.nih.gov/pubmed/28968802>; ms2lda.org/user\_guide [↑](#endnote-ref-7)
8. <http://www.jmlr.org/papers/v3/blei03a.html> [↑](#endnote-ref-8)
9. <http://eprints.gla.ac.uk/176662/> [↑](#endnote-ref-9)
10. <https://www.ncbi.nlm.nih.gov/pubmed/28968802> [↑](#endnote-ref-10)
11. <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-S7-S9> [↑](#endnote-ref-11)
12. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5521158/> [↑](#endnote-ref-12)
13. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5521158/> [↑](#endnote-ref-13)
14. <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-S7-S9> [↑](#endnote-ref-14)
15. <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-S7-S9> [↑](#endnote-ref-15)
16. <http://eprints.gla.ac.uk/147960/> [↑](#endnote-ref-16)
17. Pnas paper joe [↑](#endnote-ref-17)
18. <http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf> [↑](#endnote-ref-18)
19. <https://guides.library.uq.edu.au/research-techniques/text-mining-analysis/language-corpora> [↑](#endnote-ref-19)
20. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4550296/> [↑](#endnote-ref-20)
21. <https://www.seas.upenn.edu/~cis520/lectures/LDA.pdf> [↑](#endnote-ref-21)
22. Blei’s article [↑](#endnote-ref-22)
23. <https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd> [↑](#endnote-ref-23)
24. <https://www.sciencedirect.com/topics/neuroscience/statistical-inference> [↑](#endnote-ref-24)
25. <http://times.cs.uiuc.edu/course/598f16/notes/lda-survey.pdf> [↑](#endnote-ref-25)
26. <https://etc.usf.edu/clipart/42700/42755/cubes-25_42755.htm> [↑](#endnote-ref-26)
27. <https://www.seas.upenn.edu/~cis520/lectures/LDA.pdf> [↑](#endnote-ref-27)
28. Blei’s article as well as init\_vb from ms2ldaviz repo [↑](#endnote-ref-28)
29. <https://www.mathworks.com/help/matlab/ref/psi.html> [↑](#endnote-ref-29)
30. <https://www.pnas.org/content/113/48/13738> [↑](#endnote-ref-30)
31. Mike Cohn Book, chapter 4, user stories applied [↑](#endnote-ref-31)
32. <https://books.google.co.uk/books?id=NUW_5rs6K-wC&pg=SA8-PA3&dq=space-time+tradeoff&hl=en&sa=X&ved=0ahUKEwiN7N7miNnjAhX9QEEAHWdIAQIQ6AEIKjAA#v=onepage&q=space-time%20tradeoff&f=false> [↑](#endnote-ref-32)
33. <https://www.researchgate.net/profile/Wei-Tek_Tsai2/publication/258969772_Regression_Testing_in_an_Industrial_Environment/links/540669170cf2bba34c1e4827/Regression-Testing-in-an-Industrial-Environment.pdf> [↑](#endnote-ref-33)
34. slack [↑](#endnote-ref-34)
35. Postgres + pg restore [↑](#endnote-ref-35)
36. Specific code section used vb step, init vb, e step [↑](#endnote-ref-36)
37. <https://books.google.co.uk/books/about/Test_driven_Development.html?id=CUlsAQAAQBAJ&source=kp_book_description&redir_esc=y>

    <https://pythonspot.com/matplotlib-scatterplot/>

    IGNORE FROM HERE ON

    First of all we choose an experiment such as 190 (mass binned 5) after consultation with client. It has 500 topics, 27923 words and 2132 docs.

    **select** **count**(\*) **from** basicviz\_mass2motif **where** experiment\_id=190

    the above yields 500 (topics)

    **select** **count**(**distinct**(feature\_id)) **from** basicviz\_featureinstance

    **where** document\_id **in** (**select** id **from** basicviz\_document d **where** experiment\_id=190)

    yields 27923 (words)

    **select** **count**(**distinct**(id)) **from** basicviz\_document d **where** experiment\_id=190

    yields 2132

    len unique words yields 27923

    when we selected a document we used the following query

    **select** fi.document\_id, fm2m.probability

    **from** basicviz\_featureinstance fi, basicviz\_featuremass2motifinstance fm2m

    **where** fm2m.featureinstance\_id = fi.id

    **and** fi.document\_id **in** (**select** id **from** basicviz\_document d **where** experiment\_id=190)

    from the result 269323 was selected

    and checked it in sql if it is in the experiment:

    **select** \* **from** basicviz\_document **where** id = 269323 and experiment\_id=190

    **select** **count**(\*) **from** basicviz\_featureinstance **where** document\_id = 269323

    yields 637 just like the corpus\_data.csv file for a single document and is equivalent to len(corpus\_dict[0])

    len(unique\_topics) yields 500 which is expected of our experiment

    alpha values have been checked to be in order in accordance to the topics, the following correspond by position

    for i in range (10):

    print(alphas.items()[i])

    print(alpha\_vector[i])

    for beta we know that

    beta\_pre\_pivot[1000] gives

    [12, 12790, 0.0333402023123277]

    If we check row 12 and column 12790

    =SUBSTITUTE(ADDRESS(12,12790,4),1,"") gives us RWX2 for the specific column

    The intersection cell in excel (13, RWX2) should give us 0.0333402023123277

    =COLUMN(INDIRECT("RWX2")) gives us 12790 (just to double check)

    pivot\_table[12,12790] => 0.0333402023123277

    import csv

    with open('beta\_not\_normalised.csv') as file:

    reader = csv.reader(file)

    reader =list(reader)

    print(reader[12][12790])

    3.334020231232769776e-02

    Seems to correspond for unnormalized beta

    pivot\_table\_normalised[12][12789] -> 1.185512487090735e-08

    pivot\_table\_normalised[12][12790] -> 0.03952523801852075

    pivot\_table\_normalised[12][12791] -> 1.185512487090735e-08

    approx. close for normalised as well beta is safet therefore

    original theta tested against the database to check

    for original phi:

    # CHECKED VERSUS:

    # select fi.document\_id, fm2m.mass2motif\_id, fi.feature\_id, probability

    # from basicviz\_featureinstance fi, basicviz\_featuremass2motifinstance fm2m

    # where fm2m.featureinstance\_id = fi.id

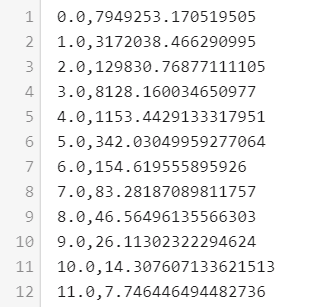
    And fi.document\_id = 269323

    Also count(\*) yields 1226 and a few values have been tested in tandem.

    Gamma is a bit big, ask the client

    Had issues with the dict not being in order alpha and beta, ofc it was unpredictable.

    Blabla csv shows that gamma is converging now which si a good thing, tested in the csv form, plot gamma\_diff as well and it was fine

    Choose another document now

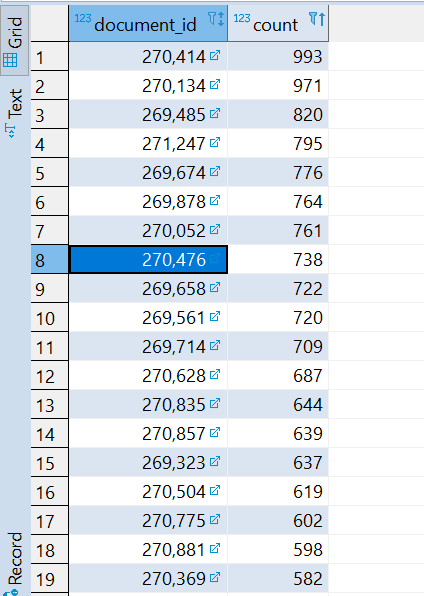
    270414

    **select** document\_id, **count**(\*)

    **from** basicviz\_featureinstance

    **where** document\_id **in** (**select** b.id **from** basicviz\_document b **where** b.experiment\_id=190)

    **group** **by** document\_id

    e.g. that word (14212)

    [4:08 PM](https://ms2lda.slack.com/archives/DKSCNEALT/p1565363327005200)

    is only in the table once. So where is your method assigning all the probability? To which topic?

    [4:08 PM](https://ms2lda.slack.com/archives/DKSCNEALT/p1565363338005500)

    and does that other topic have probability for that word?

    [4:09 PM](https://ms2lda.slack.com/archives/DKSCNEALT/p1565363376006100)

    i.e. the original model assigns word 14212 almost entirely to topic 271. Yours doesn't. But what **does**yours assign it to? And does it make sense?

    [4:10 PM](https://ms2lda.slack.com/archives/DKSCNEALT/p1565363400006500)

    also, in the image of the gammas, which topic corresponds to which row? I.e. which is 271?

    [4:10 PM](https://ms2lda.slack.com/archives/DKSCNEALT/p1565363427006700)

    and, what is the intensity of those words?

    software tested and user evaluation, suggestions for further work

    tested the intensity if ordered correctly [↑](#endnote-ref-37)