Applications of NLP to the law

October 2019

Jarrod Li
z5169806@ad.unsw.edu.au

Abstract

The legal theory surrounding precedence has compelled courts to preserve previous judgements when deciding new cases. Developments in algorithmic techniques have enabled the decomposition of previously intractable "natural language" into analysable and queryable vector spaces. A subset of natural language processing called latent semantic analysis determines the relationship between these documents of high dimensional vector spaces and their rank reduced counterparts. The aim of this research article is to assess the degree to which current algorithmic methods of language processing contribute towards producing meaningful connections between legal judgements of similar nature.

I A general background of legal procedure

In the pursuit of a fair system to resolve disputes many countries established a fundamental branch of governance, vesting this power in the judiciary. In such countries the founding principles on which courts are established vary, determined by struggles for independence, overbearing monarchies or a drive for egalitarianism. In the United States and Australia this legal basis that the judiciary was established was labelled common law which in essence proclaims the doctrine that "like cases will have like outcomes". This principle namely, state decisis, is so robust that nearly all decisions made by judges are bound towards the preservation of past decisions rather than the abolishment of decades of legal precedent.

The problem lies when judges do not follow this idea of precedent, rather making the decision to base their judgement off of their personal opinion rather than legislative or common law. Judicial restraint remains important in enshrining the widely held consensus that the judiciary does not partake in politics, activism or discrimination. To many judges, current societal and ethical standards must be applied liberally to the law based off of founding principles usually enshrined into a constitution or bill of rights. However, there are those that disagree with these "natural lawyers", these "legal positivists" place strength in the requirement that only the legislature can enact laws and it is a judges role to interpret these laws as they were intended.

Without debate over legal theory, these ideas are put forward to as a general background to understand what this paper will explore in the following sections. Common law can be used as a general purpose "litmus test" to assess how closely these principles are followed in practise. Section II will delve into the overarching idea that we are somehow able to model the foundation of all legal argument, the English language. In section III the concept of Term-Frequency

Inverse document frequency will be introduced and logically demonstrate that it is possible model all previous legal judgements in a high dimension vector space. The ability to model these documents is of low importance if we are unable to distinguish between say; a case concerning freedom of speech and the right to bear arms. Therefore, in Sections IV, V and VI this paper will dissect the mathematics behind an NLP technique called latent semantic analysis which plays a key rose in the decomposition of excessively large vector spaces populated by terms of legal judgements. Given the vast number of cases that are ruled on in the present, this paper will also review how: given a case that has not been processed by this algorithm, what previous ruling is closest in nature to the query.

Finally we will use an implementation created from the above sections in a real use scenario and observe the output. This papers findings will be assessed and included with final concluding remarks.

II Natural Language Processing

In computer science the obsession with languages often comes down to minor differences between programming languages, however most of these languages follow sets of rules that are extremely tractable to a language computers understand. Similarly, humans have learnt to process their own natural languages that follow their own sets of rules for example synonyms are words that are of similar meaning and usually can be interchanged with each other. Natural Language Processing in this instance is seeking an answer to whether given natural language as input, can the underlying properties of the English language be used in such a way as to determine how similar texts are to each other.

In this instance Natural Language Processing leverages the links between language and legal argument; refining the behaviour of these links into some mathematical model which is completely deterministic by a computer. The whole premise of NLP is not for some black box general application of artificial intelligence that can read and understand just as humans do, but using the mathematics behind how humans and nature generally govern speech, reading and writing. Tools such as Principle Component Analysis which leverages the spectral decomposition of matrices, SVD which will be heavily used in this paper, least squares regression and many other linear algebra techniques form the basis for how computer scientists are able to explore this relatively complex area of analysis. As a matter of fact it is the usage of non-linear algebra regression techniques that formed breakthroughs within the analysis of documents and more generally data, this makes perfect sense as most real world data is not going to be realistically modelled by a linear function.

A semantic relation is a relationship between words based off of the meaning of words. In computational linguistics two major determinants of semantic relations are attributional symmetry and relational symmetry. Attributional symmetry can be likened to whether words are synonymous to each other, the degree to which two words can be substituted with each other. Relational symmetry examines whether two words are analogous or rather if two words that are dissimilar are still similar in some respect. It is of interest to have some degree of relational symmetry as certain metaphors within texts can be useful in making comparisons. This is especially true in the legal domain as metaphors help organise interpretation of legal argument and can be incorporated into legal judgements, especially dissents.[1]

Vector space models are often useful for representing large corpuses of texts, hence some measure of the degree of relational symmetry of texts based off vector space models is often useful. Current research suggests that VSMs are able to determine the relational symmetry of words to a sound degree, using 374 questions from SATs that contained analogies VSMs were correct 47 percent of the time through the use of cosine similarity measures. Considering that there are 5 options to choose from in a standard SAT multiple choice paper, completely guessing would result in a 20 percent baseline; there is some evidence to suggest that VSMs are able to determine relational symmetry to some degree. These measure do become less accurate with sparse and large data sets, however these issues are addressed with TF-IDF representations in section III and singular value decomposition in section IV respectively.[2]

Natural language processing utilises different methods of representing words, some models may attempt to preserve context and order such as n grams. Others like a bag of words model is a representation best likened to taking all the words in a text and placing them in a hypothetical "bag". This bag of words

will lose all the previous grammar properties such as context of words, order and placement. This allows applications of the multiplicity of words where the only desired property is the frequency of words. This technique will be heavily utilised within the next section concerning finding computable models of text.

III Term Frequency - Inverse Document Frequency

Often large bodies of texts will contain words that are extremely specific to the topic being examined. For example the term "algorithm" will be likely referenced at least once as it forms some part of the overarching topic being explored. Therefore a good measure of what a body of text relates to can be modelled by the statistical measure of how often words are referenced in a text. The number of words in each judgement can vary on a case by case basis so a better representation for these texts are their term frequencies divided by the number of terms in the text, otherwise known as normalising each term. In this way words that are referenced often in legal judgements can be grouped into some category such as "knife" or "weapon" in murder cases, "bylaw" or "bad faith" in corporate law or "privacy" in fourth amendment cases. This becomes especially important in later sections as a statistical measure such as term frequency is relied on to be able to determine a similarity measure between judgements passed by the court.

However, assuming that the all words have equal value in texts becomes naive when comparing texts as in some cases a word such as "plaintiff" may appear many times, in many texts. This causes a problem because we do not want to give value to words that are meaningless when determining the area of law a particular judgement is deciding. The solution to this is quite intuitive, first every text being examined have their term frequencies calculated whilst keeping track of whether a text contains a specific word. Once each text has had their term occurrence determined, the terms are weighted by the total number of texts the term occurs in. Hence, if only one judgement contains a term then that term is more valuable to that text which should "bump up" its term frequency index

as that specific term is rare and has a high statistical probability to pertain to the area of law being judged.

Forming a data structure, namely vectors, to hold all the terms over every document now becomes easy to compute. First the term frequency is determined by summing up the number of occurrences of a particular word and dividing by the number of words over the entire document or,

$tf =$ (number of occurrences of a term)/(number of terms in document).

Finally, after the term frequencies of all documents have been determined it is now possible to apply weightings to every term in every document. Following on from above, once the number of documents that a term occurs in has been determined for every term in every document the term frequency is proportionally scaled by,

$idf =$ (total number of documents)/(number of documents that contain the term).

To keep the scaling reasonable and not determined by the number of documents being assessed the natural logarithm of this expression is taken. Therefore if the number of documents that a term occurs in converges to the total number of documents, the logarithm will converge to 0 or ln(1). This results in,

$idf = ln$(total number of documents)/(number of documents that contain the term).

It becomes apparent that an understanding of basic linguistic properties and statistical features of natural language are important in the area of natural language processing. This paper will now dive into the statistical foundation behind TF-IDFs.[3]

Natural language has had a tendency to follow a statistical distribution that appears in many natural phenomena such as city populations and earthquake magnitudes. Whilst there has been no solid reasoning governing Zipf's law, it still remains important in the heuristics behind how the inverse document frequency works.[4]

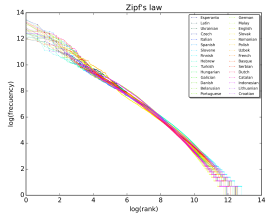


Figure 1: Rank frequency distribution of 10 million words on Wikipedia

Figure 1 is a visual representation of Ziph's law in action, it also demonstrates that not only does this power law govern the usage of words in English, but every language in existence. Suppose that some word is given the highest ranking, that is this word is used the most in a corpus of text, then by Ziph's law the second most frequent word occurs half as frequently, continuing for all words in a body of text where the n^{th} word will occur $\frac{1}{n}$ as frequently as the most frequent term.

It then stands to reason that if all texts follow this line of governance, legal texts or judgements will also be determined by some natural law governing the periodicity of terms used. This forms the basis for the intuition behind the IDF weightings of each term, less used terms such as "cafe" or "ramp" will define the circumstances that a case will be based upon. For example, some cases may fall into the broad category of negligence, however one case may reflect a situation where a cafe has poured searing hot coffee on a customer and another may be where a child has slipped on a wet ramp and broken a bone. It is therefore important to ensure these terms are not "lost" in frequency and therefore we can

discriminate between two cases on a more specific level. [Karen Spark Jones] Whilst it is of importance to understand the value the underlying rules governing our language, these often intuitive theories offer more as an argument of analogy. In the following sections this paper will aim to unravel the more rigorous mathematical proofs that will form the basis for questions governing how it is possible to compare long corpuses of text and why these methods work.

IV Singular Value Decomposition

Lawyers will spend a considerable amount of time researching outcomes of cases. Academics may also complete research through case upon case in an attempt to strengthen their studies. Judges will be given the cases cited by the defence or prosecution along with any outcomes from lower courts which in turn may themselves cite other cases. The one practise those in the legal domain will constantly require is reading; judgements, cases and opinions of the court form the bedrock for future research or argument.

Suppose there is a method of classifying these documents in a way that allows those concerned with a current area of the law to generally determine what cases are most "alike" to the circumstances of their current case. The problem lies within the nature of legal documents, the reason reading becomes tedious is the same problem as to why it is computationally difficult to compare each long TF-IDF vectors with each other, they are long.

The solution to this "curse of dimensionality" is through the use of a clever linear algebra technique called Singular Value Decomposition. We will start by forming the mathematical basis for SVD and then follow it by a more intuitive view of SVD.[5]

Consider a square matrix $m \times m$ with linearly independent eigenvectors. By the theory of linear algebra, it is known that there exists a matrix decomposition namely,

$$A = MDM^{-1}$$

where M is a matrix of column eigenvectors of A and D is a diagonal matrix with eigenvalues of A along the diagonals.

Now the importance of diagonalisation is mainly to raise A to higher powers in

a more efficient manner. This can be modelled by,

$$A^n = \underbrace{MDM^{-1}MDM^{-1}MDM^{-1} \cdot ... \cdot MDM^{-1}}_{\text{n times}}$$

$$= MD^nM^{-1}.$$

The efficient computation of n^{th} power A allows us to model long term behaviour of A, this is especially true for *leslie matrices*.

Another property of matrices is symmetry, a matrix is only symmetric if it follows the property:

$$\begin{bmatrix} a & d & \dots & e \\ d & b & \dots & \vdots \\ \vdots & \dots & \ddots & k \\ e & \dots & k & x \end{bmatrix}.$$

Symmetric matrices have a special property where their eigenvectors are orthogonally diagonalisable, however the SVD becomes easier with not just orthogonal but orthonormal eigenvectors ie. $\perp$ and $|\mathbf{v}| = 1$. Exploiting the fact that dividing each coordinate of v by the magnitude of each vector maintains orthonormality it is possible to find orthonormal eigenvectors of A.

As A becomes sufficiently large the time complexity of calculating the inverse M^{-1} of the eigenvector matrix also becomes large. This problem is only exponentiated by the typical number of words in most legal judgements.

Therefore, by finding orthonormal eigenvectors of A we have essentially cut the run time of this algorithm by a large factor as inverse matrix operations are in the order of $n^{2.x}$. This is due to the fact that if M is symmetric then it follows that $M^{-1} = M^T$, which can be viewed as a rotation matrix.

Hence, we have the following algorithm to diagonalise a matrix A:

1. Find eigenvectors and eigenvalues of A. (This step may require some help outlined in appendix A)

2. Normalise the eigenvectors to develop orthonormal eigenvectors.

3. Form the diagonal matrix with the eigenvalues of A and the eigenvector matrix M.

4. Invert M by transposition.

It is also important here to note that forming D requires placing eigenvalues in descending order with matching eigenvectors in M.

Most matrices that we will work with will not be square, intuitively this will only happen if the amount of documents that we analyse is equal to the number of terms that occur over all documents. Hence, consider an $m \times n$ matrix A. This implies that $A = [m \times n]$, now assume there exists $A^T A$ and AA^T such that $A^T A = [n \times m][m \times n] = [n \times n]$ and $AA^T = [m \times n][n \times m] = [m \times m]$. Let us now prove that $A^T A$ and AA^T are generally symmetric:

$$\text{Suppose } A = \begin{bmatrix} a & b & c \\ \ddots & \ddots & \ddots \\ d & e & f \end{bmatrix} \text{ then } A^T = \begin{bmatrix} a & \dots & d \\ b & \dots & e \\ c & \dots & f \end{bmatrix}.$$

$$\text{Then } A^T A = \begin{bmatrix} a^2 + b^2 + c^2 & ad + be + cf \\ \ddots & \ddots \\ ad + be + cf & d^2 + e^2 + f^2 \end{bmatrix} \text{ as required.}$$

Assume that for A there are two matrices, one consisting of vectors in its column space U and one consisting of vectors in its row space V. Therefore the goal is to be able to find a linear transformation A of the orthonormal basis of vectors U that maps directly to an orthonormal basis of vectors in V by some diagonal matrix Σ or:

$$AV = U\Sigma.$$

To do this we can frame the matrix representation of the linear map A as a kind of decomposition of the other three matrices, namely:

$$A = U\Sigma V^{-1} \equiv A = U\Sigma V^T$$

which we know can only be true if A itself is a symmetric, square matrix. The problem here is that A may not be square, actually most of the time A will not be square, however we have been able to prove that A^TA and AA^T are.
Now if we inspect the expansion of A^TA:

$$A^TA = V\Sigma^T UU^T\Sigma V = V\Sigma^T\Sigma V$$

which is only in terms of V, similarly for AA^T:

$$AA^T = U\Sigma VV^T\Sigma^T U = U\Sigma\Sigma^T U$$

which is only in terms of U.
Finally taking the eigenvectors of both of these matrices and orthonormalising them will result in our U and V^T, whilst the eigenvalues will be the square root of the either one of these square matrices eigenvalues. This is due to the property of the multiplication of square matrices preserving eigenvalues. We take the square root of each eigenvalue due to the fact that Σ^T and Σ are diagonal matrices,

their product will be the square each of $\sigma_i, 1 \leq i \leq n$. The last proof of SVD that will be examined by this paper will determine whether $u^T u = 0$ or whether the eigenvectors in the new basis remain orthogonal, remember $AV = U\Sigma$ must remain true. If the initially chosen eigenvectors for $A^T A$ are orthonormal, which can be assumed as the eigenvectors of a symmetric matrix are orthogonal from above, proving that $u_1^T u_2 = 0$ holds true for any U,

$$(\frac{Av_1}{\sigma_1})^T(\frac{Av_2}{\sigma_2})$$

$$= \frac{v_1^T A^T A v_2}{\sigma_1 \sigma_2}$$

as v_2 is the second eigenvector of $A^T A$ its linear transformation will result in the second eigenvalue scalar multiple of $A^T A$ which is σ_2^2 hence,

$$\frac{v_1^T v_2 \cdot \sigma_2}{\sigma_1} = 0.$$

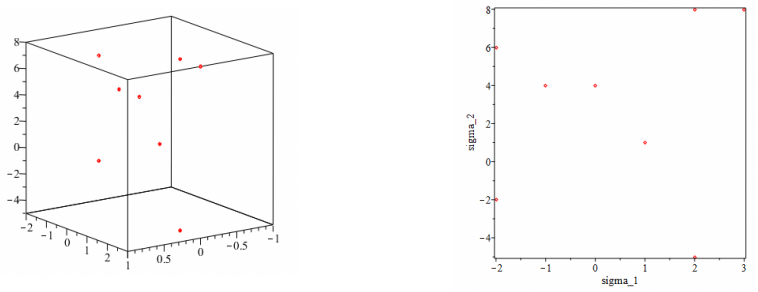
Therefore the linear map to the column space of A must also be strictly orthonormal.[6] From here taking the first term of the spectral decomposition $u_1\sqrt{\sigma_1}v_1^T$ we can determine the "first order approximation" of A. For each value $k, u_k\sqrt{\sigma_k}v_k^T$ becoming closer to the n^{th} singular value will result in a closer approximation of A similar to the Taylor series.

This idea forms the fundamental basis for Principal Component Analysis and the more intuitive application of SVD to large corpus' of texts, Latent Semantic Analysis. SVD does, however, leave open the question of whether the "k^{th} order approximation" will is the closest approximation to the original matrix.

V Low-Rank Approximation

Singular Value Decomposition works on the premise that by taking the k largest singular values; the resulting matrix reduced to the k^{th} dimension will be the k best approximation to the original matrix. More intuitively by taking the SVD of a document and lowering the number of features that will be incorporated in the approximated model of the text, how can we be sure that there will not be a better approximation by another rank k matrix?

This idea is similar to linear programming problems where questions are often framed as given a problem to maximise, find the best values for that problem maximising the result. A similar issue exists for these maximisation problems, will the solution find a global maximum? Of course in this instance it has been proven that every local maximum is a global maximum therefore ensuring the problem is feasible.



(a) Three dimensional vector space

(b) Rank reduced vector space

Figure 2: A visual example of rank reduction

It is possible to show that the new rank reduced matrix A_k is the rank k best approximation by the Eckart-Young-Mirsky Theorem: given an approximation A_k, $||A - B|| \geq ||A - A_k||$. If follows on that for any B such that $rank(B) = k$, there can be no better matrix approximation B than A_k.[7]

Remember that the Frobenius norm of a matrix is given by:

$$||A||_F^2 = |a_{1,1}|^2 + |a_{1,2}|^2 + ... + |a_{1,n}|^2 + ... + |a_{m,1}|^2 + ... + |a_{m,n}|^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} |a_{ij}|^2. \tag{1}$$

Also, the Frobenius norm can also be represented as a sum of the singular values of A, recall that the trace of a matrix ,

$$||A||_F^2 = trace(A^T A)$$

as proven in the previous section A can be decomposed into three independent matrices $U\Sigma V^T$ therefore,

$$= trace(U\Sigma V^T V \Sigma^T U^T) = trace(U\Sigma\Sigma^T U^T) = trace(\Sigma\Sigma^T U^T U)$$

$$= trace(\Sigma\Sigma^T)$$

transpositions of a diagonal matrices preserve the position of values on the diagonal and therefore the multiplication of two diagonal matrices with the same value results in squared eigenvalues,

$$= \sum_{i=1}^{min(m,n)} \sigma_i^2. \tag{2}$$

Then it must follow that by (1), $||A - A_k||_F^2 = |a_{1,k+1}|^2 + ... + |a_{1,n}|^2 + ... + |a_{k+1,n}|^2 + ... + |a_{n,n}|^2$ as the matrix A will be a square symmetric matrix and every term before $k+1$ will cancel out. Thus the Frobenius norm of the resulting matrix can be represented as a sum, $\sum_{i=1}^{n}\sum_{j=k+1}^{n} |a_{ij}|^2$ and so by the singular

value equivalency in (2),

$$||A - A_k||_F^2 = \sum_{i=k+1}^{n} \sigma_i^2$$

so proving that $||A-A_k||_F^2 = \sum_{i=k+1}^{n} \sigma_i^2 \leq ||A-B||_F^2$ will ensure that the there exists no better approximation to A_k .

Suppose A is the sum of two matrices X and Y then it follows that by Weyl's inequality,

$$\sigma_{i+j-1}(X+Y) \leq \sigma_i(X) + \sigma_j(Y)$$

and by letting $j = k+1$, $X = A - B$ and $Y = B$ holds,

$$\sigma_{i+k}(A) \leq \sigma_i(A-B) + \sigma_{k+1}(B), 1 \leq i \leq n-k. \tag{3}$$

As $rank(B) = k$ the eigenvalues between $[k+1, dim(B)]$ are equal to 0 as these vectors form the null space of the matrix, recall that we are strictly seeking the *rank k* best approximation to A and therefore the null space be of size $n-k$. The null space of a vector space is defined as all vectors $x, x \in \mathbb{F}$ such that $Ax = 0$, hence any vector in the null space will be an eigenvector with eigenvalue 0.

Applying this to the (3) results in,

$$\sigma_{i+k}(A) \leq \sigma_i(A-B)$$

as $\sigma_{k+1}(B) = 0$ as required.

It then follows that,

$$||A-B||_F^2 = \sum_{i=1}^{n} \sigma_i^2(A-B)$$

$$\geq \sum_{i=k+1}^{n} \sigma_i^2(A) = ||A-A_k||_F^2$$

where $\sigma_i(A-B)$ denotes the i^{th} eigenvalue of matrix $A-B$. Therefore, the rank k approximation A_k will always be the best approximation to A which is what makes it possible to reduce the dimension of each documents corresponding vectors.

Over the past two chapters, this paper has attempted to lay the groundwork for the mathematical foundation of the natural language processing technique called latent semantic analysis. Indeed, whilst this remains important in determining whether any applications of this algorithm will work, these types of rigorous methods do not attempt to explain how they apply to natural language in general. In the following sections a better understanding of how the mathematics connects with linguistics will be explored.

VI Latent Semantic Analysis and Indexing

Latent Semantic Analysis[8] leverages the semantic relations between words in texts explored in part III along with the idea that a similar distribution of words in texts will result in a similarity of *meaning*. This idea is usually framed as the "distributional hypothesis"; it is often argued that only focusing on the distribution of words within a text is not a sound comparison measure. Although treating texts as a "bag of words" overlooks the structure of a text, similarity will almost always preclude placement, rather words infer meaning and meaning is derived from the choice of language used in lieu of the structural placement of those words. [9]

Recall the method of modelling documents as vectors in section III, to be able to compare multiple documents this model must be improved slightly. Combining the converted document vectors together into a single matrix will group columns by document and rows by term. This can be visually represented by a series of column document vectors $\mathbf{d}_i$,

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_3 & ... & \mathbf{d}_n \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

or alternatively a series of row term vectors $\mathbf{t}_i$,

$$\begin{bmatrix} \cdots & \mathbf{t}_1 & \cdots \\ \cdots & \mathbf{t}_2 & \cdots \\ \cdots & \mathbf{t}_3 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \mathbf{t}_5 & \cdots \end{bmatrix}.$$

Now, suppose that there is a A that contains these vectors that is pairwise indexed,

$$A = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \cdots & x_{0,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix}$$

therefore each document vector is referenced by $\mathbf{d}_i^T = \underbrace{\begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,n} \end{bmatrix}}_{\text{occurrence of a term in document i}}$

and similarly for term vectors $\mathbf{t}_i = \underbrace{\begin{bmatrix} x_{1,j} & x_{2,j} & \cdots & x_{m,j} \end{bmatrix}}_{\text{occurrence of a term j in each document}}$.

Next, recognise that the SVD uses the fact that AA^T and A^TA are in terms of U, V and are square matrices, this can be applied to LSA. The matrix multiplication AA^T will take the dot product of term vectors $\mathbf{t}_i$ in both A and A^T, therefore the resulting matrix will not only contain the singular eigenvectors of U, but also will only consider the terms in the sample space. This holds true for A^TA except will examine the dot product of the document vectors $\mathbf{d}_i$. Therefore applying the SVD to this matrix will result in the spectral decomposition of A where,

$$A = U\Sigma V^T$$

expanding the singular values, left and right singular vectors of A based off of the previous assertions of A^TA and AA^T holds,

$$A = \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \cdots \begin{bmatrix} \\ \mathbf{u}_m \\ \\ \end{bmatrix} \end{bmatrix} \times \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_m \end{bmatrix} \times \begin{bmatrix} \begin{bmatrix} & \mathbf{v}_1 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & \mathbf{v}_n & \end{bmatrix} \end{bmatrix}$$

The interesting outcome of basing the right singular vectors off of the documents

of A is that now each row vector of V will correspond to a single document vector $\mathbf{d}_i$. This vector can no longer be interpreted as a comprehensible concept such as term frequency as directly as before, whilst the SVD does rely on term frequency; each value within V is now the degree to which that document is unique. This uniqueness factor is what can be used to determine how much of a particular "feature" or "idea" that particular document represents, in legal analysis these features could be tractable to a concept such as an area of the law or a constitutional protection.

Recall that it is possible to only preserve the k most significant singular values and their corresponding left and right singular vectors. The resulting matrix will be the best rank k approximation to the original matrix, preserving the most significant features of each text. Applying this will be the final direct operation on this spectral decomposition of A and all ideas in the remainder of this paper rely on the information that can be extracted from these rank k approximations.

It is possible to determine the degree to which two documents are similar, cosine similarity is based off of the cosine angle formed by two vectors in d dimensions. If two documents are deemed to be similar their vectors coordinates in euclidean space will be in approximately the same direction.

Call to mind the cosine similarity formula,

$$cos(\mathbf{d}_i, \mathbf{d}_j) = \frac{\mathbf{d}_i \cdot \mathbf{d}_k}{||\mathbf{d}_i||||\mathbf{d}_j||}.$$

In the next chapter a more formal correlation of cosine similarity to the LSA will be mentioned, describing the use in a practical application. It does, for now, suffice to say that any two documents can have their relationship measured based off of cosine similarity.

Every year the Supreme Court of the United States sits and hears cases from

all over the country. If, for instance, the LSA was used to examine the 2016 courts judgements and a vector space of those documents was created, it would be highly useful to be able to compare these judgements being passed against previous rulings. As mentioned in section I, it is important that the judiciary in common law countries such as the United States respects previous decisions and maintains stability in the judiciary as many other lower courts look to their precedent, interpretation of laws should not move with the wind so to speak. By using the cosine similarity measure, comparing new judgements against settled law is quite easy. Of course, naively this would take $O(n^2)$ operations in total, however it would be possible to significantly drive the number of comparisons down as it is known how similar two documents are to each other from the start of execution.

Another even more useful application of LSA, that is not used in later chapters, is the idea of "folding in" new texts. Once the spectral decomposition of a series of texts has been completed it is not possible to incorporate new terms of new texts into the created vector space, this would require recomputing the SVD of each text. Therefore, whilst it is possible to find coordinates of new texts in this vector space, it is not possible to include terms that may be important to those texts and therefore they must rely on the variety of terms already previously examined. "Folding in" new texts may also be useful in applications to areas of highly settled law, constitutional law comes to mind in this case. This is due to the variety of cases already heard paired with the strict number of amendments in the bill of rights. This could be incorporated in future revisions of legal analysis tools.

VII A pragmatic application to the law

Lawyers often have a rather simple yet tedious task; as judges are left to interpret the law, lawyers must best fit these interpretations to cases at hand. This endeavour becomes even narrower as judgements often set the tone for proceedings as judges are compelled to preserve previous rulings and the rulings of higher courts. It them becomes evident that a lawyers job is not inherently a theoretical one, often this is framed as jurisprudence which governs a more philosophical reasoning of the law. Indeed, lawyers must understand how cases apply, however it is the initial reading and seeking that will place the most burden on their work.

The whole premise of these final few sections is to determine whether the techniques previously mentioned will apply to more practical applications. The theory outlined in previous sections has been used in the formation of a software application that takes in the standard format of Supreme Court judgements (pdf files), completes a Latent Semantic Analysis of each judgement and outputs a csv file with the cosine similarity between every judgement in a courts annual term.

It is important to understand why US Supreme Court judgements were used as a data set for this application of LSA. The constitution gives extremely strict rules in the separation of the judiciary from executive and legislature giving good unbiased (to a certain degree) data. The US is also a common law country so the principles of stare decisis are upheld. Finally as TF-IDFs are based off of word frequency; standardising the words used or better yet the judges that write these words is of importance. As judges on the Supreme Court serve life terms, the amount of variance in terminology is kept to a minimum.

It is also useful to mention the idea of stop lists, whilst the inherit idea of TF-IDFs is to keep unrelated and overused words to a minimum; stop lists are

used as a complement to this procedure rather than a substitute. The main idea is that there are some words used almost without reason, for example the word "the" does not really give any intrinsic meaning to a text, rather being syntactically or functionally used. These common words were predefined and the program was developed to ignore any references to these words.

The results of this program will now be discussed.

VIII Results

To first test whether the ideas in this paper generally work, LSA was applied to a small set of specific, hand picked judgements. In this way some judgements were chosen to be quite similar to each other to test the viability of this program. It should also be noted that these judgements were mostly made up of "landmark cases" which revolutionised society in some way. Of course, this would not be too useful in practical applications, however as a benchmark it will suffice to see how this program acts upon predetermined data.

Judgements were chosen from three categories:

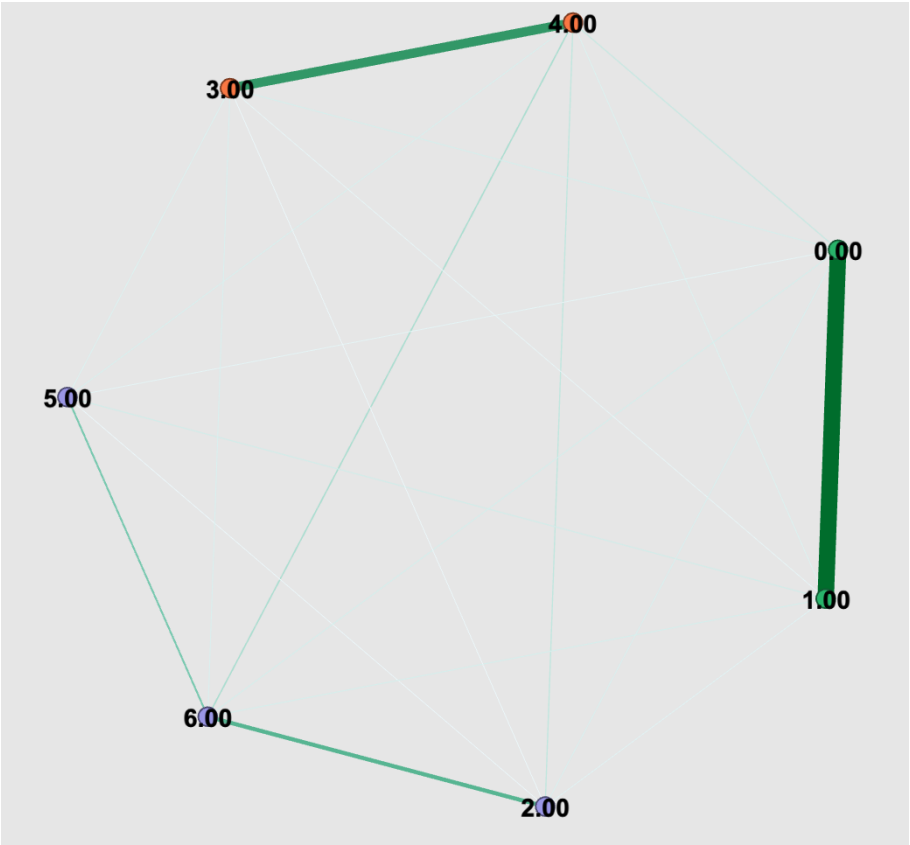
- Sexual orientation, ie. LGBT rights.
- Abortion, ie. the right to choose.
- Fourth Amendment rights, ie. unreasonable searches and seizures.

Of these categories the cases chosen were:

Sexual Orientation	Abortion Rights
2: Obergefell v. Hodges	0: Whole Woman's Health v. Hellerstedt
6: United States v. Windsor	1: Burwell v. Hobby Lobby Stores, Inc.

Fourth Amendment Rights
3: Riley v. California
4: Carpenter v. United States
5: Murphy v. National Collegiate Athletic Association

The output of this csv file was used as input into Gephi, an open source data visualiser, which resulted in the following graph:



In the above figure, the shades of green denote how "similar" two judgements are to each other and nodes are coloured based off of the area of law they represent. Interestingly, nearly every single judgement was strong connected to their correct corresponding cases. The only exception being Murphy v. National Collegiate Athletic Association.

Murphy v. National Collegiate Athletic Association is a case that was heard due to the federal government restricting state gambling laws. Whilst it could be argued that this case is based off of the states protections of an individuals

private affairs, falling under the Fourth Amendments right to undue search or seizures, this case was not decided based off of this idea. The Tenth Amendment was enacted to stop the federal government from becoming too powerful which lead to the prevention of federal government from commandeering state enacted laws.

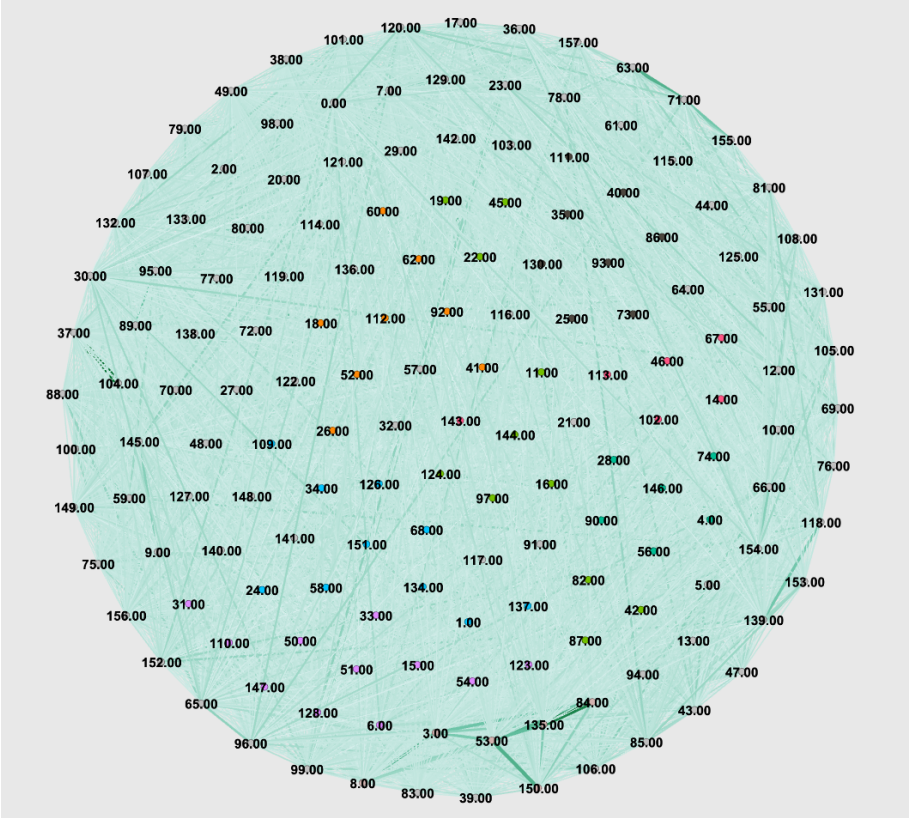
The judgement that this program deemed to be most akin to Murphy was United States v. Windsor which itself was also a case that struck down a federal law which did not recognise the union between same-sex couples. As two people could be married under New York state law, the Supreme Court deemed this law unconstitutional furthering the divide between state and federal law. Clearly, these two cases are quite similar and therefore the results using these seven cases were quite successful.

A more practical application would be to analyse two terms of Supreme Court case law then determining what similarities between cases are drawn from these cases. However, computing the singular values for such a large data set of around 150 cases is too difficult due to the sparsity and length of the TF-IDF vectors.

To find eigenvalues within a reasonable time limit that is placed on the system, a different package was used from here on out. Scipy's sparse linear algebra package finds matrix decomposition of a matrix given some constraint on the number of eigenvalues to produce. This package is not new, it is based off of ARPACK or the ARnoldi PACKage which was developed as a solution to the problem of determining the eigenvalues of matrices that are usually computationally exhaustive to solve based off of the SVD methods used above.[10] As an eigensolver this package is useful in finding the spectral decomposition as the data set used became larger and larger, from this point on any usage of the SVD to produce results will rely on a python implementation of ARPACK, not the

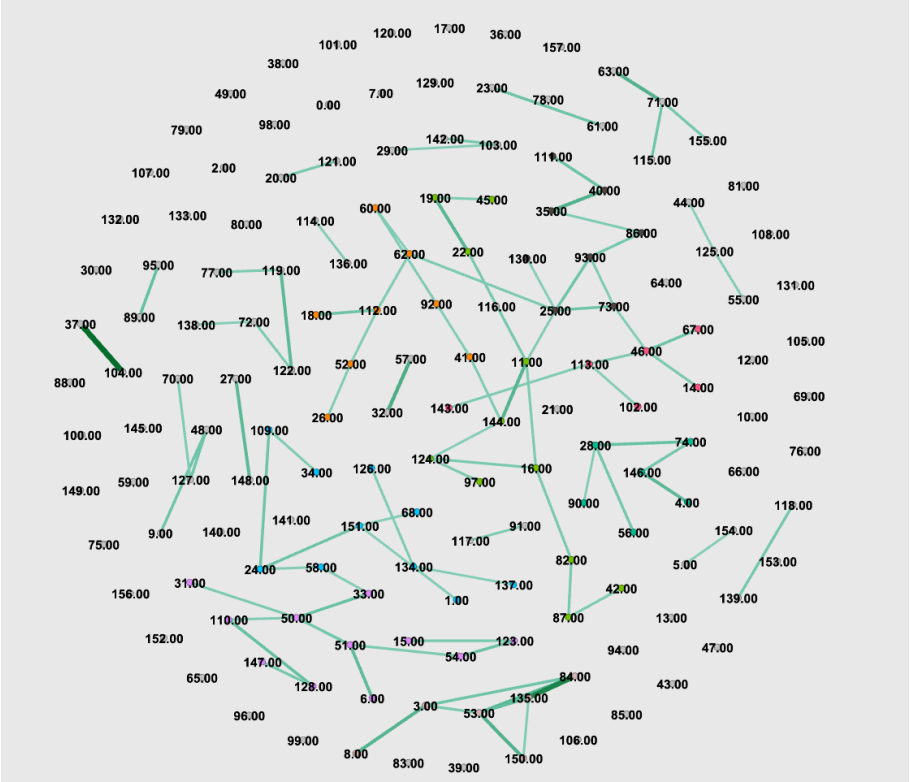
traditional SVD with rank reduction. Appendix B will explore how the Arnoldi iteration works which is core to how ARPACK is able to find eigenvalues in minimal time.

Again, graphically representing the results resulted in the following graph using the same attributes as above:



however this graph is too "crowded" with data.

Taking only the top percentile of heavy edges in the graph results in:



Now this paper will inspect some different sub graphs within this, now readable, graph and interpreting results.

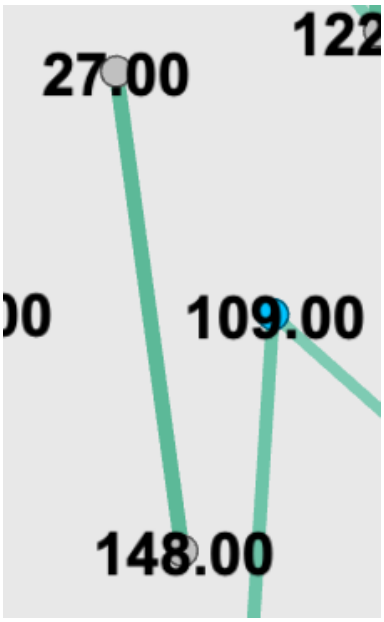


Figure 3

In figure 3, the cases of interest are 27 and 148. Judgement 27 is Abbott v. Perez 585 U.S. and judgement 148 is North Carolina v. Covington 585 U.S.. Both of these cases are predicated on racial gerrymandering.

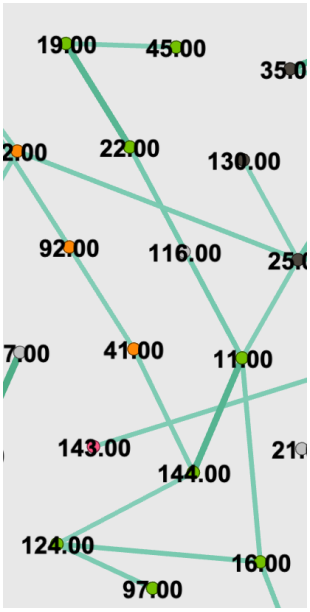


Figure 4

In figure 4, the cases of interest are 19, 11 and 45. Judgement 19: The American Legion v. American Humanist Association 588 U.S. decided that a government agency using the "cross" symbol for fallen World War I solders did not violate the establishment clause of the constitution. Judgment 11: Trump v. Hawaii 585 U.S. was predicated on a travel ban enacted by the President of the United States of mostly Muslim countries, the court found that the President did not violate the establishment clause among the plaintiffs contentions. Judgement 45: United States v. Haymond 588 U.S. is one example of a similarity found between cases that are not similar, this judgment upheld that excessive fines violated the Eighth Amendment. This case is not similar in most respects, for example it covers areas of criminal not civil law. It then can be concluded that whilst there is some basis for the similarity between judgements found; a more exhaustive list of cases is required.

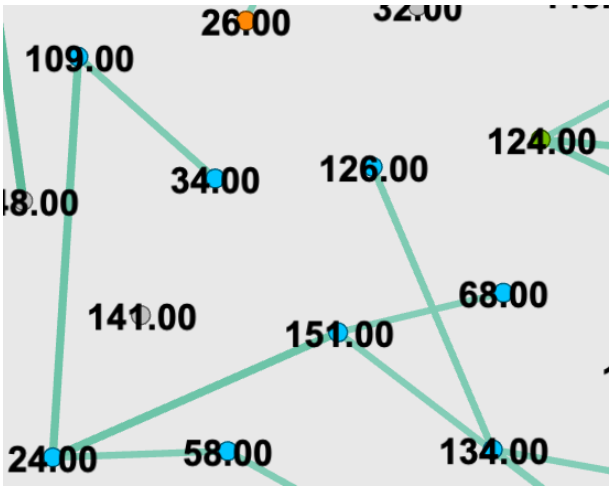


Figure 5

In figure 5, the cases of interest are 24, 58 and 109. Judgement 24: Masterpiece Cakeshop, Ltd v. Colorado Civil Rights Commission 584 U.S. is a First Amendment case based off of the compelling of protected religious speech. Judgement 58: Lozman v. City of Riviera Beach, Florida 585 U.S. determined that probable cause is not a valid reason for dismissing a claim of retaliatory arrest under the First Amendment. Judgement 109: Gill v. Whitford 585 U.S. was based off of a case of unconstitutional gerrymandering. However, in Justice Kagan's concurring opinion she noted that the plaintiffs could raise an infringement of their First Amendment protection to freedom of association. All explore wholly, or in part, some infringement upon protected First Amendment rights.

It is clear from these three examples that there is a solid grounding between the use of LSA and related techniques in the determination of the similarity of legal judgements based off of their semantic properties. Increasing the size of the data set should increases the precision of similarity measures, especially with more similar examples of judgements and situations.

IX Conclusion

Ultimately, the fundamental mathematical and linguistic basis of NLP techniques that this paper examines is key in discovering ideas and concepts in large and usually incomprehensible data sets.

Practical usage of NLP techniques in the future within the area of legal analysis has shown positive results with further work possible as outlined in this paper. By studying and combining the foundations of language, mathematics and statistical analysis with computational analysis, there is a strong basis within natural language processing to warrant further discovery of similarities between legal judgements unbeknownst to us now.

This paper concludes that whilst it is not completely definitive that results will continue to improve from the outlook outlined in section 8; there is strong empirical evidence to suggest that larger data sets will continue to provide fruitful results.

Appendices

A The Gram-Schmidt Process

This appendix is included to attempt to cover the more general ideas of this paper more thoroughly. In section IV a general theorem was used to determine that for every symmetric matrix, there exists a orthonormal decomposition of that matrix. Although this is true, these matrix decompositions may result in repeated eigenvalues, therefore this paper must explore some method of creating an orthogonal basis for this vector space to be able to completely use the SVD. In section VIII there was also a change in how the SVD was computed, as completing the full SVD was unfeasible under time constraints. This appendix will also provide a foundation for an iterative algorithm in appendix B which dramatically decreases the running time of the algorithm, providing better results in determining the value of LSA upon larger data sets.

First, assume that there exists a matrix A that has a column space of independent vectors. The Gram-Schmidt process[11] is a procedure similar to Gaussian-Elimination, however instead of aiming for an upper triangular matrix; Gram-Schmidt aims to take these independent vectors of A and transform them into orthogonal vectors.

Suppose that A contains two arbitrary vectors $\mathbf{a}$ and $\mathbf{b}$ then the goal is to find two other vectors namely A and B such that they are orthogonal projections of the original vectors. Also, to preserve an orthogonal basis over A, both A and B are to be normalised. More generally, Gram-Schmidt is seeking a solution to the general expression $A = QR$ where Q is the orthogonal basis and R an upper triangular matrix, this is also known as the QR decomposition of A.

Requiring $\mathbf{a}$ and $\mathbf{b}$ to be orthogonal can be phrased as an easier question: if $A = \mathbf{a}$ what suitable choice for B is there such that $B \perp \mathbf{a}$? It is quite intuitive

to see from here that B must be the in-place shift of $\mathbf{b}$ of the projection of $\mathbf{b}$ on A which is given by $B = \mathbf{b} - proj_{\mathbf{a}}(\mathbf{b})$. There also exists a formula for orthogonal projections that can be used in this instance,

$$P = A(A^T A)^{-1} A^T$$

which has a trivial proof, the identity matrix is formed in the centre and one is left with AA^T.

Applying this orthogonal projection holds,

$$B = \mathbf{b} - \frac{A^T \mathbf{b}}{A^T A} \cdot A$$

which is orthogonal to A as $A^T B = 0$ or formally,

$$A^T B = A^T (\mathbf{b} - \frac{A^T \mathbf{b}}{A^T A} \cdot A)$$

$$= A^T \mathbf{b} - \frac{A^T A}{A^T A} \cdot A^T \mathbf{b} = 0.$$

This process can continue for any $rank(N)$ matrix, with each vector remaining orthogonal to each other. However, as stated previously it would be necessary to find a Q such that the column space of Q is not just orthogonal but orthonormal. Hence,

$$\mathbf{q}_i = \frac{\mathbf{x}}{||\mathbf{x}||}, x \in A.$$

It also must be noted that whilst this mathematical process works wonders in theory, computationally this algorithm is numerically unstable. This is due to round off errors in floating point precision which prevent all vectors from being orthogonal to each other. The solution to this is quite trivial, however is essential and is called the Modified Gram-Schmidt algorithm.

Consider the original Gram-Schmidt process,

$$\mathbf{u}_i = \mathbf{v}_i - proj_{\mathbf{u}_1}(\mathbf{v}_i) - ... - \ proj_{\mathbf{u}_{i-1}}(\mathbf{v}_i)$$

then stabilising this algorithm is recalculating each $\mathbf{u}$ iteratively,

$$\mathbf{u}_k^{(1)} = \mathbf{v}_i - proj_{\mathbf{u}_1}(\mathbf{v}_i)$$

$$\mathbf{u}_k^{(2)} = \mathbf{u}_k^{(1)} - proj_{\mathbf{u}_2}(\mathbf{u}_k^{(1)})$$

continuing for all $\mathbf{u} < i$.

In this way the iterative procedure ensures that at every step each vector is orthogonalised against the rounding errors introduced in the previous iteration. As each successive iteration is orthogonal to the previous it holds that the j^{th} iteration will remain orthogonal to every iteration preceding it.[12]

B The Arnoldi Iteration

In Appendix A, the iterative procedure of transforming vectors that are mutually independent into orthogonal vectors has an unforeseen benefit. As a change of basis, Gram-Schmidt spanned the same vector space, however if for instance one became bored of writing out each iteration of the process; this process could stop and each computed vector would still remain orthogonal to each vector before it. This is a similar process to the Arnoldi iteration, however instead of establishing orthogonal vectors, this process maps a matrix to a Heisenberg matrix.[13] Formally this process will find,

$$AQ = QH$$

where Q are orthonormal matrices and H is a Heisenberg matrix which turns out to be tri-diagonal and symmetric if A is a Hermitian matrix. As $H = Q^{-1}AQ$ and $QQ^T = I$, $H = Q^TAQ$ where H must be symmetric.

At the heart of the Arnoldi iteration is a basic process which tries to orthogonalise a set of basis vectors which is useful as it is well known that orthogonal basis' make it easy to interchange a vector space between basis'. It remains quite useful to normalise these orthogonal vectors, providing an more stable set of basis vectors. The Krylov basis is what the Arnoldi iteration sets out to orthonormalise, this paper will now attempt to briefly explain what the Krylov basis is and what applications it has towards forming a Hermitian matrices.[14] The Krylov subspace is a span of vectors seeking a solution to $A\mathbf{x} = \mathbf{v}$ given by,

$$K_n(A, \mathbf{v}) = \text{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, ..., A^n\mathbf{v}\}$$

as it is known that the dimension of this subspace cannot exceed the dimension of the original space so n is bounded by $dim(A)$ or $K_n \subseteq K_N$. This can be

easily shown by multiplying a Krylov subspace by a constant term larger than the dimension of the original vector space,

$$AK_n = \text{span}\{A\mathbf{v}, ..., A^{n+1}\mathbf{v}\} \subseteq K_{n+1} = K_n$$

therefore for sufficiently large n, namely A_N will be an invariant subspace of span(A). Remember that an invariant subspace respects $T(A) \subseteq A$.
Now it suffices to say that Arnoldi will compute the k Krylov subspaces as an orthonormal change of basis of A. As it is known that this process is looking for an orthonormal basis Q, each q_i be the i^{th} Krylov matrix vector or,

$$K_n = \text{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, ..., A^n\mathbf{v}\} = \{q_1, q_2, q_3, ..., q_n\}.$$

The following algorithm is a standard used for the Arnoldi iteration:

1: $b = R(0, 1)$ ▷ Take a random vector from a normal distribution.
2: $q_1 = b/||b||$ ▷ Normalise the chosen vector.
3: **for** $i \leftarrow 1$ to n **do** ▷ Run the Gram-Schmidt algorithm.
4: $v = Aq_n$
5: **for** $j \leftarrow 1$ to i **do**
6: $h_{jn} = q_j v$
7: $v = v - h_{jn} q_j$
8: $h_{n+1,n} = ||v||$
9: $q_{n+1} = v/h_{n+1,n}$ ▷ Normalise the found orthogonal vector.

At termination this algorithm has produced the following,

$$H = Q^T AQ$$

by iterating n times and orthonormally projecting A onto each Krylov standard basis vector in each iteration.

Recall that the problem with the Singular Value Decomposition is a reliance on finding every single eigenvalue of AA^T or A^TA, the Arnoldi iteration would be useless if it completely reduced down to the SVD, therefore even though H_n is a projection of A, this is not what is used in popular eigenvalue generators such as ARPACK. Note that the eigenvalues of H are generally linked to the eigenvalues of A as they are invariant under a change of basis, therefore these eigenvalues are called the Ritz values of K_n usually denoted by θ_i.

Instead, ARPACK uses what is called the Implicitly Restarted Arnoldi Iteration which is quite a complex idea to understand. Questions such as will the Ritz estimates converge over time and stopping criteria are quite mathematically challenging to prove true. This paper will briefly explain the concept to illustrate how not why this method works.

The above algorithm for computing the Heisenberg matrices is again used, however this algorithm is stopped after a certain amount of time. This amount of time is governed by a factor of k iterations by which time the largest Ritz value in this matrix would have grown in proportion to the k^{th} exponent of the Krylov subspace, which is generally exceedingly fast for quite a low number of iterations. The Arnoldi iteration chooses b as a random vector sampled from a normal distribution, however this is not desirable. Once a b has been used to form a Heisenberg decomposition and an eigenvalue is deemed to be under a suitable error, the next b is chosen to in a direction away from every other vector before it. In this way each successive generation of eigenvalues will not work on the same Krylov subspace forming different results. Thus, using this method the Implicitly Restarted Arnoldi Iteration can solve eigenvalue problems such as finding singular values much faster than computing each singular value

and then rank reducing, it could be framed that this algorithm rank reduces during execution and therefore only has to complete a minimal number matrix operations.

References

[1] Peter D. Turney. Similarity of semantic relations. [On the electrodynamics of moving bodies]. *Computational Linguistics*, 32(3), 2006.

[2] Peter Turney and Michael Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60, 09 2005.

[3] Karen Jones. Jones, k.s.: A statistical interpretation of term specificity and its application in retrieval. journal of documentation 28(1), 11-21. *Journal of Documentation*, 28:11–21, 12 1972.

[4] Wikipedia. Zipf's law — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Zipf's%20lawoldid=925354295, 2019.

[5] Gilbert Strang. 18.065 matrix methods in data analysis, signal processing, and machine learning., April 2018.

[6] Gilbert Strang. 18.06 linear algebra, April 2010.

[7] G.H. Golub, Alan Hoffman, and G.W. Stewart. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its Applications*, s 88–89:317–327, 04 1987.

[8] T. K Landauer and S. Dumais. Latent semantic analysis. *Scholarpedia*, 3(11):4356, 2008. revision #142371.

[9] Magnus Sahlgren. The distributional hypothesis. *Italian Journal of Linguistics*, 20, 01 2008.

[10] R. Lehoucq, D. Sorensen, and C. Yang. Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. *SIAM*, 12 1997.

[11] Gilbert Strang. 18.06 linear algebra, April 2010.

[12] Steven Leon, Ake Bjorck, and Walter Gander. Gram-schmidt orthogonalization: 100 years and more. *Numerical Linear Algebra with Applications*, 20, 05 2013.

[13] Gilbert Strang. 18.086 mathematical methods for engineers iia, April 2006.

[14] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.