# APPLYING THE COHERENCE-BASED GENEALOGICAL METHOD (CBGM) TO THE TEXT OF THE OLD TESTAMENT: AN EVALUATION

BY

DEAN G. ELLIS

A THESIS SUBMITTED TO THE FACULTY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF DIVINITY

PROF. KENNETH CHERNEY, ADVISOR

WISCONSIN LUTHERAN SEMINARY

MEQUON, WISCONSIN

MARCH 7, 2018

## ABSTRACT

The goal of this research is to apply the Coherence Based Genealogical Method (CBGM) to the Old Testament. Deuteronomy 5 was chosen to evaluate this method. This method is being used to evaluate General Epistles in the New Testament and will result in changes to the *Editio Critica Maior* of the New Testament. To date, this method has not been applied to the Old Testament. This study relied on the development of new software algorithms to align the Hebrew text and perform the CBGM analysis. Initial results indicate that this method is applicable to Old Testament texts and is able to propose a model for the transmission of the text. Textual relationships were identified, and a proposed route of textual transmission was determined. This method has many promising applications within Old Testament textual studies. It also has several strengths and weaknesses that are addressed.

## ACKNOWLEDGEMENTS

I would like to acknowledge Timothy Wildauer, a recent graduate from Concordia University, for all his help in the design and implementation of the alignment algorithm. He was instrumental as I worked through the challenges of aligning Hebrew texts. I appreciated our Sunday evening discussions that focused on the overall challenges that CBGM presents when studying Old Testament manuscripts.

I would like to thank Professor Kenneth Cherney for all his guidance throughout this project. Finally, I would like to thank my wife and son for their support and encouragement as this project came together.

## CONTENTS

# INTRODUCTION

A new method is being used in evaluating the General Epistles in the New Testament. This new method, called the Coherence-Based Genealogical Method (CBGM), has led the publishers of the New Testament to elevate variants that were in the footnotes into the main text, and move variants that were in the main text into the footnotes. Is the CBGM method qualified to warrant these changes? Part of this thesis is aimed at evaluating this method's strengths and weaknesses as it assesses textual transmission. This method will be applied to the Old Testament.

This is the first time this method will be applied to the Old Testament. Before this method is evaluated it will be necessary to look at a snapshot of current Old Testament textual criticism. Currently, the Old Testament manuscripts are categorized as either belonging to the Masoretic and Proto-Masoretic texts, the Pre-Samaritan and Samaritan Pentateuch manuscripts, the Old Greek manuscripts, or to a group of manuscripts that don't fit anywhere. Will the CBGM method support these categories or propose another set of categories that works better? During this study, we will strive to evaluate these groupings.

One of the goals of this project is to evaluate the relationship between a manuscript and the text contains. The assumption is that an ancient text may very well be present in a manuscript that was recently copied. For instance, a late manuscript, like Leningradensis B19A, can embody an early form of the text, and vice versa, if "late" means "centuries removed from the author."

Another goal is to propose the text flow of these manuscripts using CBGM. The text flow is a diagram that shows the transmission of a text. The transmission of a text is evaluated by the similarity between the texts. This will be done by using a genealogical method.[1] A genealogical method strives to evaluate the development of a text with the aim of determining the original starting point, the arch-text. A benefit of the CBGM method is that "it allows every witness to take its own position in relation to every other one."[2] This takes us to a fundamental principle of CBGM which states, "The relationships between witnesses can be derived from the relationship of their readings."[3] This method has the potential to evaluate these relationships in the Old Testament. This will provide another tool for Old Testament scholars.

## An Overview of Current Old Testament Textual Criticism

A brief overview of current Old Testament textual criticism follows. Emmanuel Tov states that some of the goals of Old Testament textual criticism include producing a text as close to the original or restoration of the text as close as possible to the original.[4] The approach he takes is to investigate the development, copying, and transmission of the text over time and in certain geographical regions.[5] By aiming to reconstruct the original text, the arch-text also includes an analysis of the readings created over the centuries. These assumptions are key to the CBGM method and are rejected by the oral tradition school that rejects the existence of an autograph, an arch-text, in favor of multiple oral traditions that were finally written down.[4] In approaching his studies of the Old Testament text, Tov makes a distinction between the biblical text as found in

[1] Gurry, Peter J. *A Critical Examination of the Coherence-Based Genealogical Method in New Testament Textual Criticism*. (Boston: Brill, 2017), 36.
[2] Gurry, *Critical Examination*, 37.
[3] Gurry, *Critical Examination*, 39.
[4] Tov, Emanuel. *Textual Criticism of the Hebrew Bible*. Third edition, revised and expanded. (Minneapolis: Fortress 2012): 264.
[5] Tov, *Textual Criticism of the Hebrew Bible*, 265.

the Hebrew sources, or reflected in other ancient manuscripts. He recognizes the importance computer-assisted tools will have in Old Testament textual criticism. He also notes that the currently published computer databases will benefit tremendously from including historical and geographical tags alongside the text.[6] This last point is where the CBGM will deviate from previous methods used in the study of the Old Testament text. The CBGM method considers the text found in a manuscript as having the same weight as every other manuscript regardless of the known geographical or dating information. The assumption is that an ancient text may very well be present in a manuscript that was recently copied. For instance, a late manuscript, like Leningradensis B19A, can embody an early form of the text, and vice versa, if "late" means "centuries removed from the author". But, before an evaluation of the CBGM methods and principles will be undertaken, a current snapshot of Old Testament textual research will be presented.

G. E. De Rossi and Benjamin Kennicott aimed at producing a collation of Masoretic manuscripts that were evaluated alongside medieval manuscripts. This attempt at producing a collated Hebrew Bible is comfortable using predominantly eighteenth-century manuscripts.

Next, the Hebrew Bible text that is perhaps more well known, is the Kittel *Biblia Hebraica* that was published in 1936. This Bible leans heavily on the Leningrad Codex B19A as the base text. In this version there are conjectures in the apparatus.[7]

The third project is more current and widely used within our circles at Wisconsin Lutheran Seminary. The *Biblia Hebraica Stuttgartensia* (BHS) 4th Edition was published in

[6] Tov, *Textual Criticism of the Hebrew Bible*, 380.
[7] Schenker, A., ed., *Biblia Hebraica Quinta.: Deuteronomy.; Volume 18: General Introduction and Megilloth.; Volume 20: Ezra and Nehemiah.* Vol. 5. (Deutsche Bibelgesellschaft, 2011)

1977 and leans heavily on the Leningrad Codex. This codex is thought to have numerous scribal errors.[8] In general, the BHS references classes of manuscripts in the apparatus and not specific manuscripts themselves.

The fourth project is the Hebrew Old Testament Text Project (HOTTP) that is based on the BHS tradition and is also known as the *Biblia Hebraica Quinta* (BHQ), because it is intended as a fifth edition of Kittel. One major change here is that English and not Latin is used in the apparatus. The early target date for the release of this Bible was 2015. However, a more realistic release date is seen around 2020 for the full release. To date Genesis, Leviticus, Deuteronomy, Judges, Twelve Prophets, Proverbs, Ruth, Canticles, Lamentations, Esther, Ezra and Nehemiah have been released. This Bible will aim to address 5000 textual questions of special concern to the Bible translators.[8]

The fifth project is the Hebrew University Bible Project using the Aleppo Codex, which is also based on the Ben Ashera tradition base text. This project was started in 1956 and is still not complete.[9]

Finally, the *Hebrew Bible Critical Edition* (HBCE)is another project that is ongoing. This project was formerly known as the Oxford Hebrew Bible Critical Edition (OHB). This project makes use of non-Hebrew readings in addition to the Hebrew manuscripts to reconstruct the original Hebrew text. Some readings not found in the Hebrew text have been added from a retroversion of the Old Greek. This project aims to produce a diplomatic, not an eclectic, edition.

[8] Wonneberger, Reinhard. "Understanding BHS: A Manual for the Users of Biblia Hebraica Stuttgartensia". (Subsidia Biblica. Roma: Pontificio Istituto Biblico, 1990): 57.
[9] Universitah ha-`Ivrit bi-Yerushalayim. Mif`al ha-Mikra. *Textus; annual of the Hebrew University Bible Project*.( Magnes Press, 2016)

These ongoing projects to produce a new edition of the Hebrew Bible underscore different approaches in Old Testament textual criticism to reconstruct the arch-text.[10]

Of special interest in Old Testament textual criticism is the work of Drew Longacre. In his 2014 doctoral thesis entitled, "A Contextualized Approach to the Hebrew Dead Sea Scrolls Containing Exodus,"[11] he briefly evaluates the "Viability and Value of Stemmatology for the Hebrew DSS Containing Exodus."[12] He also goes on to briefly evaluate the "Reconstruction of the Documented Hebrew-Language Textual History of Exodus"[13] as well as "Iterative Relationship between Textual History and Evaluation of Variation-Units."[14] His studies recognize the value of stemmatology, a pre-requisite in starting to evaluate genealogical coherence. He also notes that the fragmentary nature of many of the Dead Sea Scrolls (DSS) pertaining to Exodus. Drew Longacre notes the difficulties in using DSS pertaining to Exodus in reconstructing the arch-text. He does make the excellent point that represetnatives of the Qumran form should be evaluated with every other manuscript. This suggestion will be addressed in this thesis as the CBGM method incorporates inter-textual comparisons as part of the method.

Although computer-assisted studies of the New Testament are receiving a lot of focus, especially in CBGM circles, the application of a computer-assisted method has not been fully implemented when it comes to the Old Testament.

[10] Fox, M.V, *Proverbs : an eclectic edition with introduction and textual commentary*. (SBL Press, 2015)
[11] Longacre, Drew. *A Contextualized Approach to the Hebrew Dead Sea Scrolls Containing Exodus*. Ph.D. (University of Birmingham, 2015)
[12] Longacre, *Contextualized Approach*, 249.
[13] Longacre, *Contextualized Approach*, 250.
[14] Longacre, *Contextualized Approach*, 251.

## The Benefits of a New Method

How does CBGM work? Peter Gurry in his book *A Critical Examination of the Coherence-Based Genealogical Method in New Testament Textual Criticism* sets out to explain the principles behind this method. This is extremely important as up until recently this method has been described by many critics as a black box. In this book, Peter Gurry explains the principles and procedures behind the method. The breakdown of the CBGM method follows. Genealogy is established by evaluating the percent agreement between two manuscripts. If the percent agreement is high enough then there is a genealogical relationship between the manuscripts. This approach becomes much more complex when trying to reconstruct the genealogy of multiple witnesses at the same time. This is where the CBGM, as a computer-assisted algorithm, comes into play.

The CBGM method makes four initial assumptions. It assumes the scribe made copies with fidelity.[15] Secondly, it assumes that the scribe used an additional source when he strayed from the main source. Thirdly, it assumes that a scribe used few rather than many sources[16]. Finally, it assumes that the witnesses are closely related to all the sources that were used. With these four assumptions in hand an overview of the method is provided below.

The first step is for the Bible scholar to draw up as many local stemmata as possible for the corpus of the text being analyzed[17]. The local stemmata are the variants within texts. The corpus of the text is the alignment of all texts. This alignment is made by aligning the witnesses and then evaluating variants that are considered by the editor to be significant. Of the significant

[15] Gurry, *Critical Examination*, 41.
[16] Gurry, *Critical Examination*, 42.
[17] Gurry, *Critical Examination*, 43.

variants the editor proposes the relationship between the variants. These relationships can be modified as the CBGM analysis continues.

The second step uses the proposed stemmata to compare the ratio of the prior and posterior readings between two witnesses.[18] The prior and posterior readings are determined when the editor evaluates all the variants. The ratio between the direction of all the variants are used to suggest which of the two witnesses is the ancestor and which is the descendent. This helps suggest a flow for the texts.

The third step determines the smallest number of ancestors to explain the text in the proposed descendant. As multiple ancestors will be present for each descendent, the challenge is to identify the key ancestors in the formation of the descendant. In short, this is done by using the smallest number of ancestors that will explain the witness that is identified as the descendent.

The fourth step evaluates the ancestors for each witness and then combines all of them to make a large map of the flow of the text. This is often referred to as the global stemma. The global stemma also suggests the flow of the text through the witnesses and identifies the arch-text.

Throughout the description of these four steps the term stemmata were introduced. A stemma represents a relationship between manuscripts. Three stemmata are identified in the CBGM. The first is the local stemma, which represents the relationship at the variant reading level. The next is the substemma which represents the relationship between a manuscript and its descendants. The final stemma is the global stemma, which is the overall representation of the relationships between all the manuscripts.

[18] Gurry, *Critical Examination*, 41.

This relationship has been identified up until this point as the agreement between the manuscripts. Manuscripts that agree more closely with one another have a stronger relationship. A suitable synonym that can be used is coherence.[19] In the CBGM method there are two types of coherence that are identified.

The first type of coherence is pre-genealogical coherence. This is the most important type of coherence.[20] Using this coherence, the relationship between two witnesses is determined based on their agreement with one another. Pre-genealogical coherence is defined by Gurry as the agreement between two witnesses at all points of comparison. This is often given as a percent agreement based on the total number of places the texts agree. This type of coherence simply evaluates how closely related two texts are. While this type of coherence shows how close two witnesses are, it does not suggest which of the manuscripts is the ancestor and which is the descendant. It does not suggest which text came first. In short, it gives coherence but not genealogical coherence. It does show that witnesses are related without giving a substemma, a text flow.

This is where genealogical coherence comes into play. While pre-genealogical coherence says that two witnesses are related, genealogical coherence gives the editor more information about the relationship between the witnesses. Genealogical coherence proposes the flow of the texts. Text will either propose a text flow or suggest that there is no direct relationship between the texts. To complicate matters, genealogical coherence may also indicate that a text has multiple ancestors.

[19] Gurry, *Critical Examination*, 49.
[20] Gurry, *Critical Examination*, 51

This ability to not only look at the relationship between witnesses but also to reconstruct their relationships gives the CBGM method an immense potential when it comes to studying the Old Testament. So how did this method come about?

## A Brief History of the Development of CBGM

The development of this method began in the 1970s when Gerd Mink at the *Institut fur Neutestamentliche Textforschung* (INTF) applied computer-based text analysis methods to the New Testament. In 1982 Mink addressed the work of Adolf Martin[21]. Mink mentioned that an evaluation of stemmata might be possible if applied to the text and not specific manuscripts. In the initial studies Mink struggled to group manuscripts into families and ended up using an iterative process to determine the genealogy of the texts under consideration.[22] From 1982 to 1997 Mink worked on his genealogical methods. The first such method that introduced coherence was with his work in his first version of ECM.[23] An increase in the number of manuscripts resulted in a second version of ECM in which the concept of coherence was introduced to readers. This work published in 2002 introduced the readers to pre-genealogical, genealogical, and stemmatic coherence.[24] These three types of coherence will be further explained in the methods section. The CBGM method was born out of this work.

## A Brief Comment on the Analysis of Variants in the Old Testament and Verbal Inspiration

[21] Gurry, *Critical Examination*, 10.
[22] Gurry, *Critical Examination*, 11.
[23] Gurry, *Critical Examination*, 13
[24] Gurry, *Critical Examination*, 15.

So far, the potential of CBGM has been discussed as it evaluates the agreement between texts and compares their variants. During the celebration of the 500th year of the Reformation, the song, “God’s Word is Our Great Heritage”, is on the lips of many of the people in our congregations. At the same time the verbal inspiration of Scripture continues to be attacked on many fronts. One such attack is based on the argument that the variant readings in Scripture somehow weaken the case that the Scriptures are verbally inspired.

The discussion concerning variants in Scripture is not only present in seminary classrooms but overflows into the congregational life. This happens when new translations of the Bible are released on the pretense of being more closely related to the original, the autograph, rather than reflecting contemporary or common language. Professor Siegbert Becker described the great damage a misunderstanding of variants can cause when he said, “We run a grave risk of playing into the hands of Bible-doubting churchmen if we refuse to take the variant readings seriously or to deal with them honestly.”[25] Undervaluing variants can cause a crisis of conscience. Placing too much value on variants may also lead to doubts regarding God’s Word. The key is to approach variants with an attitude of humility combined with common sense. Professor John Brug states, “There are really no good rules or canons for Old Testament textual criticism which are applicable to all cases. Common sense, aided by experience and good judgement in evaluating all the factors involved in each variant produces better results than a rigid application of rules.”[26]

Many opponents of God’s Word claim that variants weaken Scripture in the following ways. These opponents claim that some of the variants indicate that parts of God’s Word have

[25] Becker, Siegbert W, “Verbal Inspiration and Variant Readings,” Wisconsin Lutheran Quarterly 71 (1998): 180.

**[26]** Brug, *Textual Criticism of the Old Testament*, 6.

been lost over the centuries. They also claim that variants equal an uncertainty as to what God said in his Word. Both attacks may be addressed with the confidence that God has preserved many passages in his Word that speak to specific doctrines. When manuscripts containing similar passages are lined up it is usually clear when a variant does not belong. If nine manuscripts containing the same verse say one thing and the tenth manuscript has a word added or removed, then most of the manuscripts speak for the whole. This is a point Pieper brings out as he refers to many false teachers in the church who have claimed that Scripture is somehow incomplete because of variants.[27] In the case where few manuscripts exist, the key is to evaluate the impact of the variants on the meaning of the passage. Quite often no distinction in meaning is made. When the meaning is changed by a variant then the context of the surrounding passages is evaluated.

Using variants to attack the inspiration of Scripture therefore often has another purpose. This is nothing short of an attempt to take God's Word out of people's hands. This would be a reversal of one of the blessings of the Reformation. Professor Siegbert Becker again identifies the real issue with using variants to attack the inspiration of Scripture when he says, "Our problem with Bible-doubting Lutherans does not rest on the variant readings, but with their denial of what is plainly said in Scripture in words which the variant readings do not call into question."[28] Those who want to attack the Scripture simply misuse variants as one tool to make their case. When variants don't serve their purpose then other approaches are used.

The question that is raised is, "Are variants incompatible with the inspiration of Scripture?" The answer is that variants are not incompatible with the inspiration of Scripture.

---

[27] Pieper, Francis D. *Christian Dogmatics*. Saint Louis (Missouri: Concordia Publishing House, 1950): 240-241.
**[28]** Wisconsin Lutheran Quarterly 71 (1998): 183.

This is a key point and makes us stand in awe as we consider how God has preserved his Word over the centuries. But how can we say that variants have no impact on the inspiration of Scripture? The essence of God's Word is found in the meaning that is passed down. God's Word consists of the meaning passed down and not the outer shell of the words and syllables. This means that the New International Version published in 2011 which a catechumen studies closely while preparing for class is God's Word. It contains the meaning in the passages God verbally inspired that have been passed down through various translations.

Besides, many variants do not even impact the meaning of a word. In the case of Hebrew in the Old Testament, for example, the presence of *plene* and defective spellings do not change the meaning of the word. In other cases where words are added the meaning of the verse is not changed. If we speak of Jesus's great love, Jesus Christ's great love or Christ's great love, the thought transferred to the audience is the same. Serious Bible scholars recognize that variants are present and do not change a single doctrine of Scripture. This is quite amazing and is evidence of God's guiding hand in preserving his Word.

God assures his people that his word is inspired when he says in 2 Timothy 3:15, "From infancy you have known the holy Scriptures, which are able to make you wise for salvation through faith in Christ Jesus. All Scripture is God-breathed and is useful for teaching, rebuking, correcting and training in righteousness." We have the assurance of the verbal inspiration of Scripture given to us directly from our Savior. If this were the only verse we had regarding the verbal inspiration of Scripture then that would be enough. God in his mercy has given us many passages that point to verbal inspiration. Two examples are found in Luke 16: 27-31, and 2 Peter 1:19-21. In Luke16: 31, the importance of God's Word is emphasized when Abraham says, "He said to him, 'If they do not listen to Moses and the Prophets, they will not be convinced even if someone rises from the dead.'" God not only emphasizes the importance of his Word but the sufficiency of it as well. Likewise, in 2 Peter 1: 20-21 plainly says, "Above all, you must understand that no prophecy of Scripture came about by the prophet's own interpretation. For prophecy never had its origin in the will of man, but men spoke from God as they were carried along by the Holy Spirit." This is a marvelous passage the gives us some insight as to how God inspired his holy Word.

We marvel at how God preserved his Word over the centuries. As we look at the transmission of manuscripts we recognize that the copyists and printers were not inspired. They simply, and faithfully, copied God's Word. When the faithful copyists made errors while copying the verbal inspiration of Scripture was not called into question. In summary, variants in Scripture do not make us question verbal inspiration. When we look at the transmission of Scripture as a whole we are given a marvelous view of how God has preserved his Word over the centuries. To God be the glory.

# METHODS

## The Need for Software for CBGM Analysis

To accomplish the task of applying CBGM to the Old Testament there is a need to develop the software to assist in performing this analysis by aligning texts and then performing the iterative steps in the analysis with the help of an editor and preferably editors.

Logos Bible Software is used in this study to extract the facsimiles of the digitized manuscripts. One of the many benefits of using this software to extract digitized facsimiles is found in the ability to export selected chapters and verses.

Python and R programming languages were selected to create the software for the following reasons: Python runs across all operating systems, is open source and is the most familiar language to the author. Within the scientific community and the linguistics community there is a vast amount of support. Python version 2.7.14 was selected for the development of the software program as it handles the Hebrew text encoded in UTF-8 characters more effectively than Python version 3. Python performs all the upfront handling of the manuscripts that are exported from Logos Bible Software. The Hebrew is transliterated into the Latin alphabet for further processing for use in the R programming language. After the alignment is calculated the Hebrew characters are then restored.

R takes the manuscripts that have been processed by Python and performs the alignment. R was chosen to align the texts as it is geared towards statistical analysis and has a range of libraries supporting text analysis. This speeds up the process of realigning the manuscripts when needed. Manuscripts are commonly realigned when the percent agreement variable is changed.

By default, the program considers any word that has a 75% agreement a match. This value can be adjusted down while analyzing heavily fragmented manuscripts.

The alignment allows the user to visualize the initial alignment. An analysis of the initial alignment allows the user to see the alignment visually. This new alignment algorithm allows the user to align words that are closely related, while at the same time calculating a score matrix for each word. This is important as it shows the overall agreement of the alignment. Before this alignment algorithm was created the initial alignments using the Linguistic Python (LingPy) algorithms would take upwards of 16 hours to run. With this new alignment algorithm, the alignments take under a minute to run.

Of particular importance are the alignments of fragmented texts. When aligning fragments to full length manuscripts the challenge is keeping the fragmented text together while allowing for a certain number of gaps. If the gaps between words in a fragment are too large, then the alignment of the fragmented text is called into question. In such a scenario there is tremendous value in looking at photographs of the fragmented texts to support the gaps the algorithm inserts.

## Removing Pointing before Analysis

Manuscripts that witness to the text of Deuteronomy 5 were selected for analysis. Before Python can align the Hebrew text from a manuscript the pointing must first be removed. This is necessary as not all the manuscripts are pointed. In the case where two manuscripts are pointed it is possible to compare them with each other. This has the potential of providing more avenues to look for text differences between manuscripts, especially when considering *plene* and defective spellings.

## Manuscript Alignment

To align manuscripts for CBGM analysis each manuscript is aligned in a pairwise manner with every other manuscript. For the nineteen manuscripts under consideration this would produce 361 different pairs of alignments to analyze. One of the many benefits of designing the alignment algorithm from the ground up is the ability to take the steps of the alignment out of the black box to be able to visualize each step of the alignment.

When two manuscripts are aligned the words in each manuscript are compared with every other word in the other manuscript. If there is a match, then the word is given a score based on its length. Longer words have higher scores. If the words do not match, then the word is broken down into letters for further analysis. This allows the program to calculate the similarity of each word.

R then saves two comma-separated variable (CSV) files. The first file contains a visual representation of the alignment between two manuscripts. The second file lists where each manuscript matches.

The final step determines the best alignment. This is achieved by working through the alignment matrix file from the end of each manuscript. The program works backwards and looks for the sequence matches. This is best visualized using an Excel spreadsheet. For example, the program looks at the last two positions in the manuscript for a match. Working backwards in the alignment allows the algorithm to look for the diagonal line that is present when two manuscripts align. If a match is found, then it moves onto the next word. If no match is found then the program moves back and up a cell, in a square-wise manner, until a match is made. When a word-pair match is found then the program moves to the next word and begins the process.

Of particular importance is the optimization of how the algorithm handles gaps in fragmentary manuscripts. Although the algorithm tries to eliminate adding gaps, it is sometimes necessary for a complete alignment, especially when analyzing fragments.

## Pre-Genealogical Analysis

After aligning the manuscripts with each other the pre-genealogical analysis may begin. The editor begins by determining which texts are related to one another. This is done by comparing all the texts with one another to calculate the average percent agreement with each manuscript. The average percent agreement for all texts being analyzed is calculated and then may be used as a cutoff to determine if two texts have a relationship with one another. The points of variation are then evaluated by the editor.

When comparing manuscripts, the percent agreement is based on a comparison of the similar readings over all readings in the text. For instance, if eighteen words in both texts being compared match each other then this is considered one match and not eighteen matches. On the other side if eighteen words are different from each other then this is considered one mismatch and not eighteen mismatches. This means that a one-word mismatch and an eighteen-word mismatch are treated the same.

Variants caused by *plene* and defective spellings are not considered significant. Variants that are considered significant include large gaps within the text.

Emanuel Tov does not consider synonymous readings as being significant for further analysis.[29] He does include Linguistic and Content variants as being significant for variant analysis.[30] Tov's approach has been followed in this study.

The analysis of the variants by the editor is of primary importance as these decisions are used in determining the flow of the genealogical coherence of the text. Once the flow of a text is decided all the possible ancestors of a text are then considered.

## Coherence Based Genealogical Analysis

With the pre-genealogical analysis complete, the genealogical analysis is ready to run. The pre-genealogical analysis may be adjusted by the editor at any time to incorporate new insights gained through the genealogical analysis. This is important as the method itself is constantly improving.

The program takes each pair of sequences that has a percentage agreement higher than the mean and then continues to look at the overall text flow for the sum of the local stemmata. For instance, if nine variants, stemmata, are proposed by the editor to move from text A to text B while one variant is proposed to move from text B to text A, then there is a strong suggestion that text A is the ancestor and text B is the descendent. If, on the other hand, 4 passages are proposed to have come from text B while 6 from text A, then the determination of the flow of the text is not as confident.

[29] Tov, *Textual Criticism of the Hebrew Bible*, 264.
[30] Tov, *Textual Criticism of the Hebrew Bible*, 267.

## Retroversion of the Old Greek to Hebrew

A particularly challenging subsection of this thesis is the evaluation of retroverting the Old Greek into Hebrew. Using a concordance to link the lemma of an Old Greek word to the lemma of a commonly used Hebrew root is relatively simple. Putting the Old Greek into Hebrew grammar presents additional challenges.

One approach to accomplish this is to take a consensus Hebrew text from the BHS and then use a concordance to compare whether the consensus text uses the most common Hebrew words. If it does then the Hebrew form is used. If it does not then the program alerts the editor and waits for their input.

Another approach is to use a natural language toolkit, which applies the grammar of a particular language when translating. Google translate uses this approach in conjunction with artificial intelligence and deep-language learning. Deep-language learning uses artificial intelligence to learn the rules of a language. These rules are then used to translate a document from one language into another.

For the initial stages of this project Tov's retroversion from the Old Greek to Hebrew was used as the Old Greek representative[31].

[31] Tov, Emanuel. *The Parallel Aligned Hebrew-Aramaic and Greek Texts of Jewish Scripture*. Bellingham, WA: Lexham Press, 2003.

## RESULTS:

The most crucial component of the pre-genealogical analysis is the alignment of the manuscripts under consideration. A solid and reliable alignment will produce a solid and reliable CBGM output. In contrast, a weak alignment will produce questionable CBGM output. Figure 1 shows two aligned texts from Deuteronomy 5:1-10, with word matches colored from red, indicating a weak word match, to green, indicating a strong word match. A quick glance over the alignment matrix reveals a diagonal line. This line is indicative of the best text alignment match.

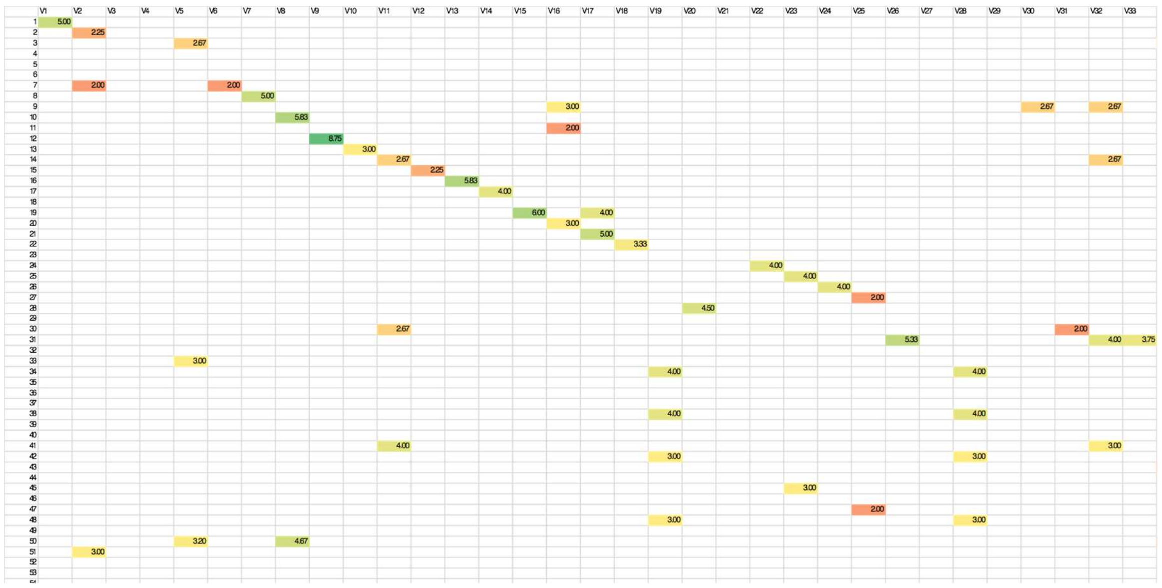


*Figure 1. Example of Text Alignment between BHS and LXX.*

Word-pairs that are not connected to each other are removed. Word-pairs that are connected to each other, forming a diagonal line, are further analyzed. Initial tests evaluated whether there were any benefits to requiring at least three word matches in a row to be considered for further analysis. This was problematic for heavily fragmented texts. Removing this requirement has not had a negative impact on the overall alignment of all the manuscripts evaluated so far. Once word-pairs that are not connected with other word-pairs are removed, the text alignment matrix produces a clean diagonal line as can be seen in Figure 2.

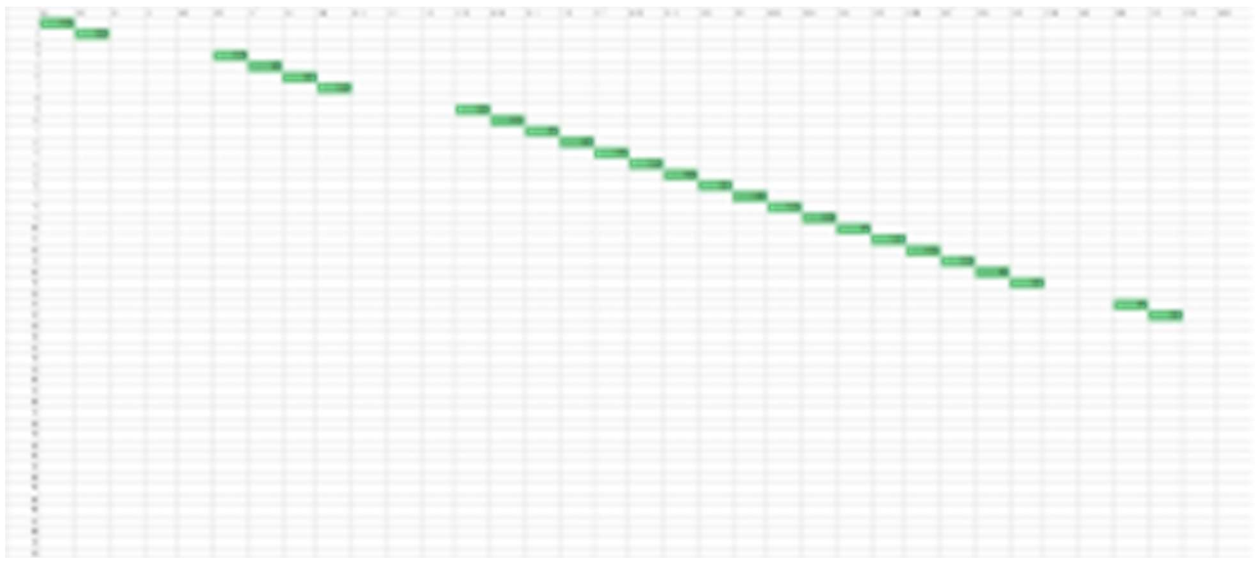


*Figure 2. Cleaned alignment matrix with all single word matches removed.*

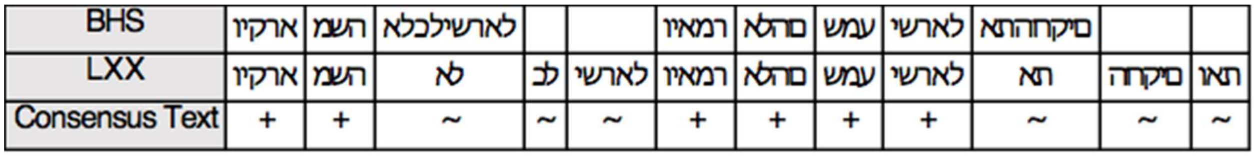
The alignment matrix is then used to determine the best possible word matches. The word matches are transliterated using a Python algorithm. A new transliteration algorithm was created to provide transparency throughout the whole analysis procedure. This algorithm can easily be adapted to reflect many transliteration algorithms in the field of Old Testament textual studies. See Appendices A and B for the code that is used to align these matches. Initial word-pair matches are given a score. The score considers each word length as well as the percent agreement between each word. An example is seen in Table 1. The benefit of viewing the word matches in the text is that it allows the program to align the Hebrew texts with one another.

| ID | BHS Text Position | BHS Word being matched (transliterated) | LXX Word being matched (transliterated) | LXX Text Position |
|---|---|---|---|---|
| 1 | 1 | gkyzb | gkyzb | 1 |
| 2 | 2 | p1f | p1f | 2 |
| 3 | 4 | gkbpz | gkbpz | 6 |

| ID | BHS Text Position | BHS Word being matched (transliterated) | LXX Word being matched (transliterated) | LXX Text Position |
|---|---|---|---|---|
| 4 | 5 | bnfo | bnfo | 7 |
| 5 | 6 | 1pt | 1pt | 8 |
| 6 | 7 | k1zbn | k1zbn | 9 |
| 7 | 9 | gb2fp1vjko | fp1vjko | 13 |
| 8 | 10 | b1z | b1z | 14 |
| 9 | 11 | brmk | brmk | 15 |
| 10 | 12 | ecz | ecz | 16 |

*Table 1. Example of transliterated word matches from Deuteronomy 5:1-10 between BHS and LXX.*

The result is an alignment between two texts. An example is seen in Table 2. The consensus text represents the relationship between the texts. The consensus text has a '+' if the two words are an exact match, a '~' if the words do not match, and a 'M' if the words are 75% similar. The agreement criteria, currently set at 75%, is useful for detecting words with *plene* and Defective spelling. This means that if a word has three out of four letters which match a second word with three letters, then there is a match. Raising the percent agreement requirement for the word-match produces a better alignment with longer texts. Lowering the percent agreement requirement for the word-match is beneficial when looking at heavily fragmented texts. The example below also shows that some words are really a group of words without a space. This is a result of the digitization of the texts.



| BHS | ויקרא | משה | אלכלישראל | | | ויאמר | אלהם | שמע | ישראל | אתהחקים | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LXX | ויקרא | משה | אל | כל | ישראל | ויאמר | אלהם | שמע | ישראל | את | החקים | ואת |
| Consensus Text | + | + | ~ | ~ | ~ | + | + | + | + | ~ | ~ | ~ |

*Table 2. Example of an alignment between BHS and LXX.*

Following the alignment of the manuscripts the pre-genealogical and genealogical coherence can be calculated. Table 3 shows the results of the alignment of five selected texts from Deuteronomy 5. Even though single words are not used to calculate the percentage agreement, the statistics are included to give an indication of the overall agreement between the manuscripts. The word matches indicate words that have 100% agreement. The similar word matches row shows words that have greater than 75% agreement. The word gaps row shows where there is no word agreement.

Table 3 shows the results from the CBGM run. The first two columns list the two manuscripts being compared. The third column shows the percentage match between the two manuscripts. The average percent agreement for all manuscripts analyzed was 23%. The standard deviation was 0.21, indicating that 95% of the results ranged from 23% to 44% agreement. The average percent agreement was used to set the criteria for determining a genealogical relationship. The fourth column lists the number of word matches. The low number of word matches is heavily dependent on how fragmented the text is. The fifth column lists the shortest manuscript in the analysis. The sixth, seventh and eighth columns show the genealogical coherence between the two manuscripts. The Ancestor (A) column shows how many variants suggested that the first manuscripts variants gave rise to the second manuscripts variants. The Descendant (D) column shows the number of variants in the text represented in the second manuscript is thought to have given rise to the text represented in the first manuscript. The Undetermined (U) column lists the number of variants that could not be classified as being either an Ancestor or Descendent.

| Text 1 | Text 2 | Percent Match | Number of Matches | Length of Shortest Sequence | A->D | A<-D | U |
|---|---|---|---|---|---|---|---|
| LXXMTParallel Dt G | BHSSESB Dt | 90% | 211 | 235 | 7 | 1 | 21 |
| QDeutN Dt | BHSSESB Dt | 84% | 198 | 235 | 8 | 3 | 19 |
| BHSSESB Dt | QPhylB Dt | 56% | 57 | 101 | 9 | 5 | 6 |
| BHSSESB Dt | QPhylJ Dt | 56% | 122 | 217 | 9 | 9 | 9 |
| BHSSESB Dt | QPhyl Dt | 53% | 53 | 100 | 11 | 3 | 2 |
| BHSSESB Dt | QDeutOFrag Dt | 50% | 2 | 4 | | | |
| QDeutJ Dt | QDeutOFrag Dt | 50% | 2 | 4 | | | |
| QDeutN Dt | QDeutOFrag Dt | 50% | 2 | 4 | | | |
| BHSSESB Dt | QPhylO Dt | 43% | 10 | 23 | 4 | 0 | 0 |
| BHSSESB Dt | QPhylA Dt | 37% | 10 | 27 | 3 | 0 | 0 |
| QPhyl Dt a | QDeutOFrag Dt | 33% | 1 | 3 | | | |
| QPhylR Dt | QDeutOFrag Dt | 33% | 1 | 3 | | | |
| BHSSESB Dt | QPhylL Dt | 31% | 9 | 29 | 3 | 0 | 0 |
| QDeutJ Dt | QPhylO Dt | 30% | 7 | 23 | 2 | 0 | 0 |
| QDeutN Dt | QPhyl Dt a | 25% | 19 | 77 | 6 | 0 | 1 |
| LXXMTParallel Dt G | QPhyl Dt a | 25% | 19 | 77 | 6 | 1 | 1 |
| QPhylJ Dt | QDeutJ Dt | 24% | 18 | 75 | 1 | 3 | 2 |
| Samaritan Pent | QPhyl Dt a | 24% | 19 | 77 | | | |
| QPhyl Dt | QDeutJ Dt | 20% | 15 | 75 | | | |
| QDeutJ Dt | QPhylA Dt | 19% | 5 | 27 | | | |
| BHSSESB Dt | QPhylG Dt | 18% | 39 | 212 | | | |
| BHSSESB Dt | QPhyl Dt a | 18% | 14 | 77 | | | |
| QPhylG Dt | QPhyl Dt a | 18% | 14 | 77 | | | |
| QPhyl Dt a | QPhylO Dt | 17% | 4 | 23 | | | |
| QDeutN Dt | QDeutJ Dt | 16% | 12 | 75 | | | |
| LXXMTParallel Dt G | QDeutJ Dt | 16% | 12 | 75 | | | |
| QPhyl Dt a | QPhylA Dt | 15% | 4 | 27 | | | |
| QPhylJ Dt | QPhyl Dt a | 14% | 11 | 77 | | | |
| QPhyl Dt | QPhyl Dt a | 14% | 11 | 77 | | | |
| QPhyl Dt a | QPhylL Dt | 14% | 4 | 29 | | | |
| BHSSESB Dt | QDeutJ Dt | 13% | 10 | 75 | | | |
| QPhylG Dt | QDeutJ Dt | 13% | 10 | 75 | | | |
| QPhylB Dt | QPhyl Dt a | 13% | 10 | 77 | | | |
| QPhylB Dt | QDeutJ Dt | 8% | 6 | 75 | | | |
| QDeutJ Dt | QPhylL Dt | 7% | 2 | 29 | | | |
| QPhyl Dt a | QDeutJ Dt | 7% | 5 | 75 | | | |
| BHSSESB Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhyl Dt a | QPhylR Dt | 5% | 1 | 20 | | | |
| QDeutJ Dt | QPhylR Dt | 5% | 1 | 20 | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| QDeutN Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhylA Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhylB Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhylG Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhylJ Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhylL Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhylO Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| QPhyl Dt | QPhylR Dt | 5% | 1 | 20 | | | |
| LXXMTParallel Dt G | QPhylR Dt | 5% | 1 | 20 | | | |
| | | | | | | | |
| **Total (N)** | **47** | **47** | **47** | **47** | | | |
| **Average** | | **23%** | **20** | **58** | | | |
| **SD** | | | **44** | **59** | | | |

*Table 3. Genealogical Coherence Parameters.*

A visual representation of the results in Table 3 are seen in Figure 3. This representation is referred to as a text flow diagram. The line weights indicate the strength of the relationship between manuscripts. The LXX and BHS have the strongest relationship. The arrow indicates that the text in the LXX preceded the text in the BHS. A dotted line indicates that the relationship between the manuscripts is weak. The relationship between the BHS and Qphyl J manuscript is weak and no coherence, text flow, could be determined with the current dataset. The text in Qphyl B, A, and L appear to have been preceded by the text of the BHS. The following point must be reemphasized. The assumption is that an ancient text may very well be present in a manuscript that was recently copied. For instance, a late manuscript, like Leningradensis B19A, can embody an early form of the text, and vice versa, if “late” means “centuries removed from the author”. The LXX and BHS appear to be more closely related to each other than to the Samaritan Pentateuch.

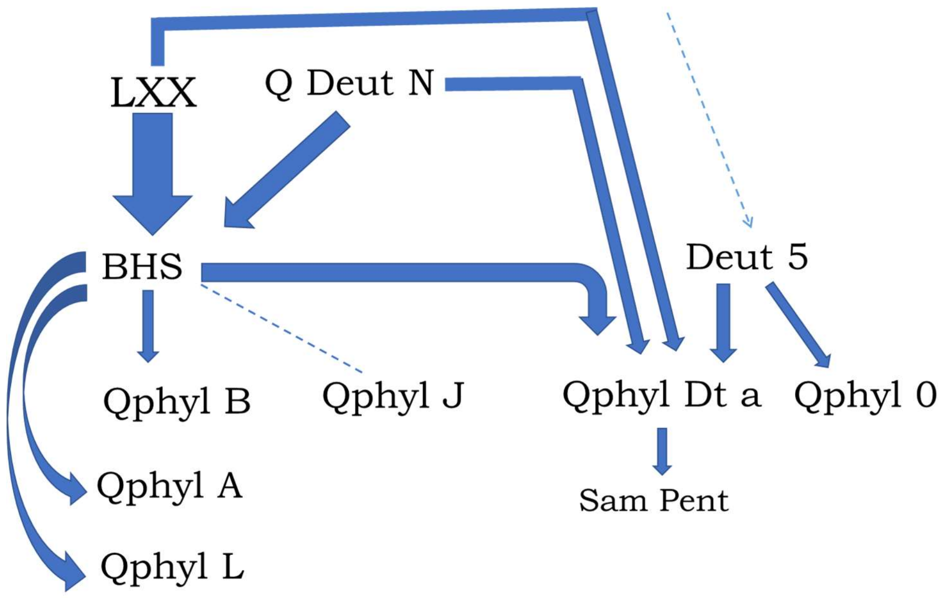


*Figure 3. Text Flow Diagram showing the overall relationship between the texts.*

# DISCUSSION

The main components relating to CBGM that will be discussed include the current state of CBGM analysis of Old Testament texts, current challenges to implementing CBGM in the Old Testament, strengths and weaknesses inherent in this method and recommendations for future research.

## The Current State of CBGM Analysis of Old Testament Texts

So far, we can align Old Testament texts using a new alignment algorithm. This allows the editor to determine pre-genealogical coherence as well as genealogical coherence. Based on the genealogical coherence the editor is then able to create a text flow diagram that proposes the overall transmission of a text from manuscript to manuscript. This will provide a novel way to assess the current groupings of the manuscripts.

## Current Challenges to Implementing CBGM in the Old Testament

What is the best way to handle fragmented texts? The algorithm attributes longer variants as ancestors and gaps in the text as indicators of being descendants. There are many situations where these rules are not valid. If the gaps in a fragmented text are caused by the physical deterioration of the manuscript, then this rule will not apply.

The current algorithm analyzes manuscript pairs. An improvement to the alignment algorithmn will include aligning all the manuscripts together to get a master alignment. This will produce a *corpus* of text would then allow the software algorithm to distinguish between variants that differ between two manuscripts and variants that exist in the corpus of text but are similar between the two manuscripts being analyzed.

The current algorithm will also benefit from applying the principle of parsimony to the texts. This is not a major focus as the number of manuscripts available for analysis is limited to begin with.

## Strengths and Weaknesses Inherent in this Method

Strengths include the ability to align Old Testament texts and determine their relationship with one another. Another strength of this method includes the use of two parameters, agreement and a sum of the variants supporting an ancestor or descendent role, to evaluate the relationship between the manuscript and ultimately the text. The fact that dating information and geographical information are excluded helps to ensure that we are investigating relationships between texts, not simply the manuscripts that represent them.

Weaknesses in the method include the calculation points that introduce subjectivity into the method. One area where subjectivity may bias the results includes the editor's input in determining whether each variant is an ancestor or descendent. To mitigate the first weakness, a more thorough statistical model may be sought to draw the line between related and unrelated manuscripts.

As a new method in Old Testament textual analysis the software that has been developed for this project also will have many opportunities to be optimized and grow. As the dataset grows further software changes may be needed to adapt to new discoveries.

Strengths and weaknesses considered, the CBGM has tremendous potential in the study of Old Testament manuscripts. It is able to combine the questions generated by the editor with computer assisted software to take a meta-genomic look at Old Testament texts. This will in turn give the editor a high-level picture of the relationship between texts.

## Recommendations for Future Research.

The first and most pressing area for future research using this method includes retroverting the Old Greek manuscripts into Hebrew. Using computer-assisted technologies will greatly speed this portion of the project up. There are a number of software approaches that may be used. The simplest approach may include using an electronic concordance to replace the Old Greek with its corresponding Hebrew word. The challenge here will include providing a word form and not just a lemma. Another approach may involve using machine learning to accomplish this task. This would entail providing the machine learning algorithm and a database of ancient texts. The benefit here is that the algorithm would be able to learn and apply common laws of grammar to the retroversion.

Another area of future research will benefit greatly from creating a database of all the pre-coherence and genealogical coherence text information. Firstly, this would prevent any reprocessing of the data at the lower levels of the program. This would also speed up any reanalysis that is needed as more texts are added. Such a database would be best served using a web interface such as WordPress attached to a freely available database.

One last area of future research that would be beneficial includes the handling of the steps between the computer analysis and the editors input. Using a web-based system would also speed up this step.

# APPENDIX A. PROCESSING AND TRANSLITERATING THE HEBREW MANUSCRIPTS IN PYTHON

```
#Initial processing of Hebrew manuscripts in a *.txt file.
import codecs,unicodedata, string, os, sys
import datetime, time


#get number of sequences
fi = codecs.open("C:\Users\ellis\Desktop\input.txt","r", "utf-8") #in
visual studio enable encodings.utf_8
c = 0
for line in fi:
    c += 1
print str(c) + ' sequences found'
fi.close()

print "Initialzling Global Variables"
num_of_sequences = c

seq = [u''] * num_of_sequences
scores = [(u'', u'', 0.0)] * num_of_sequences
alig = [(u'', u'', 0.0)] * num_of_sequences
clean_score = [(u'',0.0)] * num_of_sequences #score to baseline

#coi = ["u'\u05da'", "u'\u05ea'", "u'\u05fa'","u'\u05db'", "u'\u05eb'",
"u'\u05fb'","u'\u05dc'","u'\u05ec'","u'\u05fc'","u'\u05dd'","u'\u05ed'","u
'\u05fd'","u'\u05de'","u'\u05ee'","u'\u05fe","u'\u05df'","u'\u05ef'","u'\u
05ff'","u'\u05d0'","u'\u05e0'","u'\u05f0'","u'\u05d1'","u'\u05e1'","u'\u05
f1'","u'\u05d2'","u'\u05e2'","u'\u05f2'","u'\u05d3'","u'\u05e3'","u'\u05f3
'","u'\u05d4'","u'\u05e4'","u'\u05f4'","u'\u05d5'","u'\u05e5'","u'\u05f5'"
,"u'\u05d6'","u'\u05e6'","u'\u05f6'","u'\u05d7'","u'\u05e7'","u'\u05f7'","
u'\u05d8'","u'\u05e8'","u'\u05f8'","u'\u05d9'","u'\u05e9'","u'\u05f9'"]
coi =
["u'\u05c6'","u'\u05d0'","u'\u05d1'","u'\u05d2'","u'\u05d3'","u'\u05d4'","
u'\u05d5'","u'\u05d6'","u'\u05d7'","u'\u05d8'","u'\u05d9'","u'\u05da'","u'
\u05db'","u'\u05dc'","u'\u05dd'","u'\u05de'","u'\u05df'","u'\u05e0'","u'\u
05e1'","u'\u05e2'","u'\u05e3'","u'\u05e4'","u'\u05e5'","u'\u05e6'","u'\u05
e7'","u'\u05e8'","u'\u05e9'","u'\u05ea'"]
cod =
{"u'\u05c6'":"a","u'\u05d0'":"b","u'\u05d1'":"c","u'\u05d2'":"d","u'\u05d3
'":"e","u'\u05d4'":"f","u'\u05d5'":"g","u'\u05d6'":"h","u'\u05d7'":"i","u'
\u05d8'":"j","u'\u05d9'":"k","u'\u05da'":"l","u'\u05db'":"m","u'\u05dc'":"
n","u'\u05dd'":"o","u'\u05de'":"p","u'\u05df'":"q","u'\u05e0'":"r","u'\u05
e1'":"s","u'\u05e2'":"t","u'\u05e3'":"u","u'\u05e4'":"v","u'\u05e5'":"w","
u'\u05e6'":"x","u'\u05e7'":"y","u'\u05e8'":"z","u'\u05e9'":"1","u'\u05ea'"
:"2"}
nums = ["1", "2", "3", "4", "5", "6", "7", "8", "9"]
```

```
print "Importing Fle Objects"
fi = codecs.open("C:\Users\ellis\Desktop\input.txt","r", "utf-8")

#read the file data in
c = 0
for line in fi:
    seq[c] = line
    print "line " + str(c) + "\n" + seq[c]
    c = c + 1

print "About to remove pointing"

fo = codecs.open("C:\Users\ellis\Desktop\clean_seq.txt","w", "utf-8")

c = 0
clean_seq = [u''] * num_of_sequences
for line in seq:
    script = u''
    title = u''
    tscript = u''
    skip = False
    for char in line:
        #print "'"+char+"' ->"+ repr(char)
        if char == "]":
            skip = False
        elif char == "[":
            skip = True
            continue
            #script = script + char
            #print "Editing bracket found"
        elif skip == True:
            continue
        elif repr(char) in coi:
            #script = script + char
            script = script + char
            tscript = tscript + cod[repr(char)]


            #print "Hebrew Const Found"
        elif char == " ":
            try:
                if script[-1] != " ": #[-1] if LTR and [0] if RTL
                    #pass #Test for Pairwise analysis
                    #script = script + " " + char
                    script = script + char
                    tscript = tscript + char

                    #print "Space Found"
            except:
                pass
                #print "Space error"
        elif char in nums:
            #script = script + " " + char #TRYING GAPS
            #script = char + script
            pass
            #print "Number Found"
        elif char in string.ascii_letters:
            title = title + char
            #pass
        else:
            pass
    #fo.write(title + "\n")
    #fo.write(u'START\n')
    fo.write(title + u'="')
    fo.write(tscript + u'"\n"')
    #fo.write(script + u'"\n')
    #fo.write(u'END\n')

    title = u''
    script = u''
    tscript = u''

    #print "Clean Seq " + clean_seq[c] + "\n\n\n\n"
    c += 1

print "Pointing removed"
fo.close()
print "Done." + str(datetime.datetime.now())

#Post processing of the results

# -*- coding: utf-8 -*-

import codecs,unicodedata, string, os, sys, ast
import datetime, time


 #in visual studio enable encodings.utf_8

#coi = ["u'\u05da'", "u'\u05ea'", "u'\u05fa'","u'\u05db'", "u'\u05eb'",
"u'\u05fb'","u'\u05dc'","u'\u05ec'","u'\u05fc'","u'\u05dd'","u'\u05ed'","u
'\u05fd'","u'\u05de'","u'\u05ee'","u'\u05fe","u'\u05df'","u'\u05ef'","u'\u
05ff'","u'\u05d0'","u'\u05e0'","u'\u05f0'","u'\u05d1'","u'\u05e1'","u'\u05
f1'","u'\u05d2'","u'\u05e2'","u'\u05f2'","u'\u05d3'","u'\u05e3'","u'\u05f3
'","u'\u05d4'","u'\u05e4'","u'\u05f4'","u'\u05d5'","u'\u05e5'","u'\u05f5'"
,"u'\u05d6'","u'\u05e6'","u'\u05f6'","u'\u05d7'","u'\u05e7'","u'\u05f7'","
u'\u05d8'","u'\u05e8'","u'\u05f8'","u'\u05d9'","u'\u05e9'","u'\u05f9'"]
#coi =
["u'\u05c6'","u'\u05d0'","u'\u05d1'","u'\u05d2'","u'\u05d3'","u'\u05d4'","
u'\u05d5'","u'\u05d6'","u'\u05d7'","u'\u05d8'","u'\u05d9'","u'\u05da'","u'
\u05db'","u'\u05dc'","u'\u05dd'","u'\u05de'","u'\u05df'","u'\u05e0'","u'\u
05e1'","u'\u05e2'","u'\u05e3'","u'\u05e4'","u'\u05e5'","u'\u05e6'","u'\u05
e7'","u'\u05e8'","u'\u05e9'","u'\u05ea'"]
cod =
{"u'\u05c6'":"a","u'\u05d0'":"b","u'\u05d1'":"c","u'\u05d2'":"d","u'\u05d3
'":"e","u'\u05d4'":"f","u'\u05d5'":"g","u'\u05d6'":"h","u'\u05d7'":"i","u'
\u05d8'":"j","u'\u05d9'":"k","u'\u05da'":"l","u'\u05db'":"m","u'\u05dc'":"
n","u'\u05dd'":"o","u'\u05de'":"p","u'\u05df'":"q","u'\u05e0'":"r","u'\u05
e1'":"s","u'\u05e2'":"t","u'\u05e3'":"u","u'\u05e4'":"v","u'\u05e5'":"w","
u'\u05e6'":"x","u'\u05e7'":"y","u'\u05e8'":"z","u'\u05e9'":"1","u'\u05ea'"
:"2"}
nums = ["1", "2", "3", "4", "5", "6", "7", "8", "9"]


#get list of final files

for root, dirs, files in os.walk('C:\Users\ellis\Desktop\CBGM Alignment'):
    pass
for file in files:
    #print "root " + root
    tfile = file.split("_")
    if tfile[-1] == "final.csv":
        #print file
        fn = root + "\\" + file
        fi = codecs.open(fn,"r", "utf-8")
        fo = codecs.open(root + "\\_" + tfile[0] + "_" + tfile[1] + "_" +
tfile[-2] +"_final_h.csv","w", "utf-8")
        fi.readline()
        match = fi.readline()
        first = fi.readline()
        second = fi.readline()
        print "match " + match + "\n"
        fo.write(match)
        print "first " + first + "\n"
        print "second " + second + "\n"

        #make it happen
        script = u''
        for char in first:
            #print "'"+char+"' ->"+ repr(char)

            if char in cod.values():
                #script = script + char
                k = cod.keys()
                for i in k:
                    if cod[i] == char:
                        script = script + ast.literal_eval(i)
                #print "Hebrew Const Found"
            elif char == ",":
                script = script + ","
            elif char in nums:
                #script = script + " " + char #TRYING GAPS
                #script = char + script
                pass
                #print "Number Found"
            elif char in string.ascii_letters:
                script = script + char
                #pass
            else:
                pass
        fo.write(script+"\n")
        script = u''
```

```
for char in second:
    #print "'"+char+"' ->"+ repr(char)

    if char in cod.values():
        #script = script + char
        k = cod.keys()
        for i in k:
            if cod[i] == char:
                script = script + ast.literal_eval(i)
        #print "Hebrew Const Found"
    elif char == ",":
        script = script + ","
    elif char in nums:
        #script = script + " " + char #TRYING GAPS
        #script = char + script
        pass
        #print "Number Found"
    elif char in string.ascii_letters:
        script = script + char
        #pass
    else:
        pass
fo.write(script+"\n")
#fo.write(title + "\n")
#fo.write(u'START\n')
#fo.write(script + u'"\n')
#fo.write(u'END\n')


#done happening


fi.close()
fo.close()
```

## APPENDIX B. PAIRWISE ALIGNMENT ALGORITHM IN R

```
match_sequences = function(first_fragment_words_list,
first_fragment_letters_list, second_fragment_words_list,
second_fragment_letters_list, match_percent = 0.75)
{
  total_match_count = 0
  full_match_list = matrix(ncol = 6)
  full_match_list = as.data.frame(full_match_list)

  full_match_matrix = as.data.frame(matrix(0, nrow =
length(second_fragment_words_list), ncol =
length(first_fragment_words_list)))


  for (first_fragment_index in 1:length(first_fragment_words_list))
  {
    # for each word in the longer fragment, compare it to each word in the
shorter fragment
    second_fragment_index = 0
    repeat
    {
      second_fragment_index = second_fragment_index + 1
      if(second_fragment_index > length(second_fragment_words_list))
      {
        break
      }
      # if the words are an exact match, mark it down
      if(first_fragment_words_list[first_fragment_index] ==
second_fragment_words_list[second_fragment_index]) # compare words
      {
        total_match_count = total_match_count + 1
        full_match_list = rbind(full_match_list, c(total_match_count,
first_fragment_words_list[first_fragment_index], first_fragment_index,
second_fragment_words_list[second_fragment_index], second_fragment_index,
1 * nchar(first_fragment_words_list[first_fragment_index])))
        full_match_matrix[second_fragment_index, first_fragment_index] =
nchar(first_fragment_words_list[first_fragment_index])
      }
      # if they don't match perfectly, look at how closely they match
      else
      {
        first_fragment_word_letters =
first_fragment_letters_list[[first_fragment_index]] # first word that
didn't match
        second_fragment_word_letters =
second_fragment_letters_list[[second_fragment_index]] # second word that
didn't match
```

```
        word_letter_match_count = 0
        average_word_length =
round(mean(c(length(first_fragment_word_letters),
length(second_fragment_word_letters))))
        if(length(first_fragment_word_letters) >
length(second_fragment_word_letters))
        {
          first_word_letters = first_fragment_word_letters
          second_word_letters = second_fragment_word_letters
        }
        else
        {
          first_word_letters = second_fragment_word_letters
          second_word_letters = first_fragment_word_letters
        }
        letter_index = 0
        previous_letter_index = 0
        if(average_word_length != 0)
        {
          for (first_word_letter_index in 1:length(first_word_letters))
          {
            repeat
            {
              # if the letters match, mark it down and see if the next
letter matches
              letter_index = letter_index + 1
              if(letter_index > length(second_word_letters))
              {
                letter_index = previous_letter_index
                break
              }
              else if(first_word_letters[first_word_letter_index] ==
second_word_letters[letter_index])
              {
                word_letter_match_count = word_letter_match_count + 1
                previous_letter_index = letter_index
                break
              }
            }
          }
          # if the percentage of matching letters is greater than the
threshold, call it a match and mark it down
          if((word_letter_match_count/average_word_length) >=
match_percent)
          {
            total_match_count = total_match_count + 1
            full_match_list = rbind(full_match_list, c(total_match_count,
first_fragment_words_list[first_fragment_index], first_fragment_index,
second_fragment_words_list[second_fragment_index], second_fragment_index,
(word_letter_match_count/average_word_length) *
nchar(first_fragment_words_list[first_fragment_index])))
            full_match_matrix[second_fragment_index, first_fragment_index]
= (word_letter_match_count/average_word_length) *
nchar(first_fragment_words_list[first_fragment_index])
```

```
          }
        }
      }
    }
  }

  colnames(full_match_list) = c("match number", "first word", "first word
index",
                                "second word", "second word index",
"percent match")
  full_match_list = full_match_list[2:length(full_match_list[,1]),]
  full_match_list <<- full_match_list
  full_match_matrix <<- full_match_matrix
  # save the lists and return both of them

  #print(paste("made all matches for", sorted_names[1], "and",
sorted_names[2]))
  return(list(full_match_list, full_match_matrix))
}


clean_sequence_matches = function(full_match_matrix)
{
  # # create a new matrix as big as the previous one
  # clean_match_matrix = as.data.frame(matrix(rep(0,
  #                                        times =
length(full_match_matrix[1,]) * length(full_match_matrix[,1])),
  #                                      nrow =
length(full_match_matrix[,1]), ncol = length(full_match_matrix[1,])))

  matched_matrix = full_match_matrix[1:(length(full_match_matrix[,1])-1),
1:(length(full_match_matrix[1,])-1)]
  matched_matrix = matched_matrix +
full_match_matrix[2:length(full_match_matrix[,1]),
2:length(full_match_matrix[1,])]

  clean_match_matrix = data.frame(matrix(0, ncol =
length(matched_matrix[1,]), nrow = length(matched_matrix[,1])))



  # # new
  #
  # for(i in length(matched_matrix[,1]):1)
  # {
  #   for(j in length(matched_matrix[1,]):1)
  #   {
  #     if(matched_matrix[i,j] > full_match_matrix[i+1,j+1])
  #     {
  #       clean_match_matrix[i,j] = 1
  #       if(i > 1)
  #       {
  #         if(j > 1)
  #         {
```

```
  #           clean_match_matrix[i-1,j-1] = 1
  #         }
  #       }
  #     }
  #   }
  # }

  for(row_number in 2:length(full_match_matrix[,1]))
  {
    for(column_number in 2:length(full_match_matrix[1,]))
    {
      # walk through the rows and if the number is not a zero, check the
next word to see if they match too
      # if they do, mark both of them as consecutive matches
      if(full_match_matrix[row_number,column_number] > 0)
      {
        if(full_match_matrix[row_number-1,column_number-1] != 0)
        {
          clean_match_matrix[row_number,column_number] = 1
          if(row_number == 2 || column_number == 2)
          {
            clean_match_matrix[row_number-1,column_number-1] = 1
          }
        }
        else if(column_number < length(full_match_matrix[1,]))
        {
          if(row_number < length(full_match_matrix[,1]))
          {
            if(full_match_matrix[row_number+1,column_number+1] != 0)
            {
              clean_match_matrix[row_number,column_number] = 1
            }
          }
        }
      }
    }
  }


  # new

  clean_match_matrix[is.na(clean_match_matrix)] = 0

  clean_match_matrix <<- clean_match_matrix

  #write.csv(clean_match_matrix, file = paste(sorted_names[1],
sorted_names[2], "clean_match_matrix.csv", sep = "_"))

  #print(paste("cleaned the matches for", sorted_names[1], "and",
sorted_names[2]))

  return(clean_match_matrix)
  # # save the data frame and return it
  # clean_match_matrix <<- clean_match_matrix
  #
  # print(paste("cleaned the matches for", sorted_names[1], "and",
sorted_names[2]))
  #
  # return(clean_match_matrix)
}

final_sequence_match = function(clean_match_matrix)
{
  # create another data frame as big as the input data frame
  final_matches_matrix = as.data.frame(matrix(rep(0,
                                                  times =
length(clean_match_matrix[1,]) * length(clean_match_matrix[,1])),
                                              nrow =
length(clean_match_matrix[,1]), ncol = length(clean_match_matrix[1,])))

  final_matches_indices = as.data.frame(matrix(c(0,0), ncol = 2))

  # # new
  #
  # final_matches_matrix = data.frame(matrix(0, ncol =
length(clean_match_matrix[1,]), nrow = length(clean_match_matrix[,1])))
  #
  # i = length(clean_match_matrix[,1])
  # j = length(clean_match_matrix[1,])
  # write.csv(clean_match_matrix, file = "testing.csv")
  # while(i > 0 && j > 0)
  # {
  #   previous_row = i
  #   previous_column = j
  #   print(paste("row", i, "column", j, clean_match_matrix[i, j]))
  #   if(clean_match_matrix[i,j] == 1)
  #   {
  #     #print(paste("row", i, "column", j))
  #
  #     final_matches_matrix[i,j] = 1
  #     i = i - 1
  #     j = j - 1
  #   }
  #   else
  #   {
  #     found = FALSE
  #     repeat
  #     {
  #       previous_row = previous_row - 1
  #       previous_column = previous_column - 1
  #       if(previous_row == 0 || previous_column == 0)
  #       {
  #         i = 0
  #         j = 0
  #         break
  #       }
  #       for(row in i:previous_row)
  #       {
  #         for(column in j:previous_column)
  #         {
  #           #print(paste("in row", row, "and column", column, "I found",
clean_match_matrix[row, column]))
  #           if(clean_match_matrix[row, column] == 1)
  #           {
  #             if(row > 1 && column > 1)
  #             {
  #               if(clean_match_matrix[row-1,column-1] == 1)
  #               {
  #                 #print(paste("row", row, "column", column))
  #
  #                 final_matches_matrix[row,column] = 1
  #                 final_matches_indices = rbind(c(row, column),
final_matches_indices)
  #
  #                 i = row - 1
  #                 j = column - 1
  #                 found = TRUE
  #                 break
  #
  #               }
  #             }
  #             else
  #             {
  #               final_matches_matrix[row,column] = 1
  #               final_matches_indices = rbind(c(row, column),
final_matches_indices)
  #
  #               i = row - 1
  #               j = column - 1
  #               found = TRUE
  #               break
  #
  #             }
  #           }
  #         }
  #         if(found)
  #         {
  #           break
  #         }
  #       }
  #       if(found)
  #       {
  #         break
  #       }
  #     }
  #   }
  #   #print(i)
  #   #print(j)
  # }
  #
  row_number = length(clean_match_matrix[,1])
  column_number = length(clean_match_matrix[1,])
  next_row_number = 1
  next_column_number = 1
  while(row_number > 0 && column_number > 0)
  {
    # while the words match, save it and then check the previous word
    while(clean_match_matrix[row_number,column_number] == 1)
    {
      final_matches_indices = rbind(c(row_number, column_number),
final_matches_indices)
      final_matches_matrix[row_number,column_number] = 1
      row_number = row_number - 1
      column_number = column_number - 1
      if(column_number == 0 || row_number == 0)
      {
        break
      }
    }
    if(column_number == 0 || row_number == 0)
    {
      break
    }

    # if the words don't match, expand out in a square to see where they
begin to match again
    next_row_number = row_number - 1
    next_column_number = column_number - 1
    next_match_found = FALSE
    while(!next_match_found)
    {
      for(current_row in row_number:next_row_number)
      {
        if(clean_match_matrix[current_row,next_column_number] == 1)
        {
          if(clean_match_matrix[current_row-1,next_column_number-1] == 1)
          {
            column_number = next_column_number
            row_number = current_row
            next_match_found = TRUE
            break
          }
        }
      }
      if(!next_match_found)
      {
        for(current_column in column_number:next_column_number)
        {
          if(clean_match_matrix[next_row_number,current_column] == 1)
          {
            if(clean_match_matrix[next_row_number-1,current_column-1] ==
1)
```

```
            {
              column_number = current_column
              row_number = next_row_number
              next_match_found = TRUE
              break
            }
          }
        }
      }
      if(next_match_found)
      {
        break
      }

      # the rest of this checks to make sure it doesn't walk off the end
of the data frame
      next_row_number = next_row_number - 1
      next_column_number = next_column_number - 1
      if(next_row_number == 0)
      {
        next_row_number = 1
        if(next_column_number == 0)
        {
          column_number = 0
          row_number = 0
          break
        }
      }
      if(next_column_number == 0)
      {
        next_column_number = 1
      }
    }

    if(column_number == 0)
    {
      if(row_number != 0)
      {
        column_number = column_number + 1
      }
    }
    else if(row_number == 0)
    {
      row_number = row_number + 1
    }
  }


  final_matches_indices =
final_matches_indices[1:(length(final_matches_indices[,1])-1),]

  number_of_matches = length(final_matches_indices[,1])
  shortest_sequence = min(length(final_matches_matrix[,2]),
length(final_matches_matrix[2,]))
```

```
    percent_match <<- number_of_matches / shortest_sequence
    print(paste(sorted_names[1],sorted_names[2],"percent match",
percent_match,"number of matches",number_of_matches,"length of shortest
sequence",shortest_sequence,sep = ","))

  final_matches_matrix <<- final_matches_matrix
  final_matches_indices <<- final_matches_indices

  #print(paste("made final matches for", sorted_names[1], "and",
sorted_names[2]))

  return(list(final_matches_matrix, final_matches_indices))
}

finalCSV = function(final_results, fragment1_list, fragment2_list)
{
  matching = c()
  first_sequence_list = c()
  second_sequence_list = c()
  previous_first_sequence_word_number = as.numeric(final_results[1,1])
  previous_second_sequence_word_number = as.numeric(final_results[1,4])
  first_sequence_list = c(first_sequence_list,
fragment1_list[previous_first_sequence_word_number])
  second_sequence_list = c(second_sequence_list,
fragment2_list[previous_second_sequence_word_number])
  #print(paste(previous_first_sequence_word_number,
fragment1_list[previous_first_sequence_word_number],
fragment2_list[previous_second_sequence_word_number],
previous_second_sequence_word_number))
  matching = c(matching, "match")
  i = 0
  if(length(final_results[,2]) > 1)
  {
    for (i in 2:(length(final_results[,2])))
    {
      first_sequence_word_number = as.numeric(final_results[i,1])
      second_sequence_word_number = as.numeric(final_results[i,4])
      if(first_sequence_word_number ==
(previous_first_sequence_word_number + 1) && second_sequence_word_number
== (previous_second_sequence_word_number + 1))
      {
        first_sequence_list = c(first_sequence_list,
fragment1_list[first_sequence_word_number])
        second_sequence_list = c(second_sequence_list,
fragment2_list[second_sequence_word_number])
        #print(paste(first_sequence_word_number,
fragment1_list[first_sequence_word_number],
fragment2_list[second_sequence_word_number], second_sequence_word_number))

        matching = c(matching, "match")
      }
      else
      {
        first_sequence_difference = first_sequence_word_number - 
previous_first_sequence_word_number - 1
        second_sequence_difference = second_sequence_word_number - 
previous_second_sequence_word_number - 1
        for(j in 1:(max(c(first_sequence_difference, 
second_sequence_difference))))
        {
          matching = c(matching, "mismatch")
          if(previous_first_sequence_word_number + j < 
first_sequence_word_number)
          {
            first_sequence_list = c(first_sequence_list, 
fragment1_list[previous_first_sequence_word_number + j])
          }
          else
          {
            first_sequence_list = c(first_sequence_list, " ")
          }
          if(previous_second_sequence_word_number + j < 
second_sequence_word_number)
          {
            second_sequence_list = c(second_sequence_list, 
fragment2_list[previous_second_sequence_word_number + j])
          }
          else
          {
            second_sequence_list = c(second_sequence_list, " ")
          }
          #print(paste(previous_first_sequence_word_number + j, 
fragment1_list[previous_first_sequence_word_number + j], 
fragment2_list[previous_second_sequence_word_number + j], 
previous_second_sequence_word_number + j, "mismatch"))
        }
        first_sequence_list = c(first_sequence_list, 
fragment1_list[first_sequence_word_number])
        second_sequence_list = c(second_sequence_list, 
fragment2_list[second_sequence_word_number])
        #print(paste(first_sequence_word_number, 
fragment1_list[first_sequence_word_number], 
fragment2_list[second_sequence_word_number], second_sequence_word_number))
        matching = c(matching, "match")

      }


      previous_first_sequence_word_number = first_sequence_word_number
      previous_second_sequence_word_number = second_sequence_word_number
    }
  }

  final_final_sequence_list = rbind(matching, first_sequence_list, 
second_sequence_list)

  final_final_sequence_list <<- final_final_sequence_list
  #write.csv(final_final_sequence_list, file = paste(sorted_names[1],
sorted_names[2], "final.csv", sep = "_"))

  #print(paste("made nice csv for", sorted_names[1], "and",
sorted_names[2]))
}

match_two_fragments = function(fragment1, fragment2, names, split_sequence
= " ", match_percent = 0.75)
{
  # split each of the fragments into a list of words and a list of letters
  fragment1_list = strsplit(fragment1, split_sequence)[[1]]
  fragment1_words_list = c()
  for (i in 1:length(fragment1_list)) {
    fragment1_words_list = c(fragment1_words_list,
strsplit(fragment1_list[i], "")[1])
  }
  fragment2_list = strsplit(fragment2, split_sequence)[[1]]
  fragment2_words_list = c()
  for (i in 1:length(fragment2_list)) {
    fragment2_words_list = c(fragment2_words_list,
strsplit(fragment2_list[i], "")[1])
  }
  # figure out which fragment is longer and keep them in that order
  if(length(fragment1_list) > length(fragment2_list))
  {
    first_fragment_list = fragment1_list
    first_fragment_letter_list = fragment1_words_list
    second_fragment_list = fragment2_list
    second_fragment_letter_list = fragment2_words_list
    sorted_names <<- c(names[1], names[2])
  }
  else
  {
    first_fragment_list = fragment2_list
    first_fragment_letter_list = fragment2_words_list
    second_fragment_list = fragment1_list
    second_fragment_letter_list = fragment1_words_list
    sorted_names <<- c(names[2], names[1])
  }
  # find where the sequences match and save the results
  full_matchs_list = match_sequences(first_fragment_list,
first_fragment_letter_list, second_fragment_list,
second_fragment_letter_list, match_percent)
  full_matches = full_matchs_list[[1]]
  full_matches_matrix = full_matchs_list[[2]]
  # find sequential word matches and save them
  clean_matrix = clean_sequence_matches(full_matches_matrix)
  # walk through and find the closest matchs, and save the results
  best_matches_list = final_sequence_match(clean_matrix)
  best_matches_matrix = best_matches_list[[1]]
  best_matches_sequence = best_matches_list[[2]]
  # for each final match, print out the indices along with the words, and
save them to a data frame
  final_results = as.data.frame(matrix(rep(0, times = 4), ncol = 4))
  for (i in 1:length(best_matches_sequence[,1]))
  {
    first_word_index = best_matches_sequence[i,2]
    first_word = first_fragment_list[first_word_index]
    second_word_index = best_matches_sequence[i,1]
    second_word = second_fragment_list[second_word_index]
    final_results = rbind(final_results, c(first_word_index, first_word,
second_word, second_word_index))
    #print(paste(first_word_index, first_word, second_word,
second_word_index))
  }
  final_results = final_results[2:length(final_results[,1]),]
  names(final_results) = c(paste(sorted_names[1], "position", sep = "_"),
paste(sorted_names[1], "word", sep = "_"), paste(sorted_names[2], "word",
sep = "_"), paste(sorted_names[2], "position", sep = "_"))

  ##########print(paste("made final list results for", sorted_names[1],
sorted_names[2]))

  finalCSV(final_results, first_fragment_list, second_fragment_list)

  # Write selected CSV files to the directory

  if(percent_match < percent_match_threshold)
  {
    #print(paste("These sequences do not match closely enough. They only
match", percent_match))
    # write.csv(full_match_list, file = paste(sorted_names[1],
sorted_names[2], "full_match_list.csv", sep = "_"))
      #write.csv(full_match_matrix, file = paste(sorted_names[1],
sorted_names[2], "full_match_matrix.csv", sep = "_"))
      if (percent_match > 0.23)
      {
          write.csv(full_match_matrix, file = paste(sorted_names[1],
sorted_names[2],"grey_area","full_match_matrix.csv", sep = "_"))
          write.csv(final_final_sequence_list, file =
paste(sorted_names[1], sorted_names[2], "grey_area","pmt", percent_match,
"final.csv", sep = "_"))
          #print(paste("In the grey area between the average (0.23) and
the upper SD (0.44) ", percent_match))
      }
  }
  else
  {
      #print(paste("These sequences match", percent_match, "for agreement
calculation ", percent_match))

    matches <<- matches + 1

    #write.csv(clean_match_matrix, file = paste(sorted_names[1],
sorted_names[2], "clean_match_matrix.csv", sep = "_"))
```

```
    write.csv(final_matches_matrix, file = paste(sorted_names[1],
sorted_names[2], "final_matches_matrix.csv", sep = "_"))
    #write.csv(final_matches_indices, file = paste(sorted_names[1],
sorted_names[2], "final_matches_indices.csv", sep = "_"))

    #write.csv(final_results, file = paste(sorted_names[1],
sorted_names[2], "final_list_results.csv", sep = "_"))

    write.csv(final_final_sequence_list, file = paste(sorted_names[1],
sorted_names[2],"pmt",percent_match, "final.csv", sep = "_"))


  }


}

match_all_sequences = function(list_of_fragments, list_of_fragment_names)
{
  for(fragment1 in 1:(length(fragments) - 1))
  {
    for(fragment2 in (fragment1 + 1):length(fragments))
    {
      total_comparisons <<- total_comparisons + 1
      beginning_local_time = proc.time()
      names = c(fragment_names[fragment1], fragment_names[fragment2])
      match_two_fragments(fragments[[fragment1]], fragments[[fragment2]],
names, " ", 0.65)
      total_local_time = proc.time() - beginning_local_time
      #print(total_local_time)
    }
  }
  #print(paste(matches/total_comparisons, "of the comparisons matched"))
}


matches = 0
total_comparisons = 0


QPhylDta = "gkyzb p1zbn g1pt fp1vjknpe2o b2o nt1mn2 kfgfo cfgcbhtprg
czmnrg icizcl nko cvrko eceko mk nb iczk2f2gl fb1 nb tnk2kfgf tfgf mkkn2o
pvrfkf nl bnfko bnb 2tgceo mk bz nbpz bnl vs nb1o tn vrbnfkl bn1f nlye tgq
bk nb kr2l mk ckgomk kfmn 2pgrfkoc1pko 1n1ko gtk1b b2 1n zctko 1pg 2t1f nb
21b2 kpko 2tzl bz k1f bnfkl nb2 mk tce fkpl mb1zipge btk 1c2 n2l mpglbcklb
2tk2 cbzw kfgf bnfkl ztkl gnb xgl kfnl nb 2zxi g"
QPhylRDt = "gkyzbp1zbng1pt k1zbnb2fiykogb2fp1vjkgcbhnpe2ob2ont1tprgczcizcl
nmn2kfgficzk2fmnrgikocvrkoekfgftocf2glf1b tpklc
kfgfgckrkmoct2ffkpnfb2eczkfgfmkkn2opvrb1nbtnk2znbpz bkkfcekomk
nbfkfnlbnfkobotnvr1fnlvsnn2pgrfc1pko pptngb1zcbzwp2i2gb1zcpkop2i2nbzw
nb212igfnfognb2tgceomkbnfklbyetgqbon1kogtnzctko n1grbk
gtg1fisenbnvkonbgfckn1gpzkpxg2k nb21knbkrzk1bb21pg n1gb
1pgzb2kgof1c2nye1gmb1zxglkfgfbnfkl 112kpko2t2l mkckgotk1c2nfbnfklnb2t1f
```

```
mnpnbmfb2fgcrlgc2lgtcelgbp2lg1gzlgipzlgmncfp2lgdzlb2lmpgl2mktcefk2cbzw
pxzkogkgxkblkfgfbnfklp1ockeihyfgchzgtrjgkftnmqxglkfbcklgb2plmb1zkfgfbnfkl
nptqkbzkmgqkpklgnptqkkjcnltnfbepfb1zkfgfbnfklrg2qnl nb2zxiipgebtklgnb
22bgfck2ztkl1efggtceggbp2g1gzggipgzggmnb1znztkl "
QDeutJDt = "gkyzb pg1f bn mgn k1zbnkgbpz o 1pf k1zbn b2 figyko gb2 fp1vjko
b1z br egcz crkmo fkgo fhf gnpe2o b2o 1pz2o nt1g2fgf bnz2 tprg czk2 cigzc
nb krg mz2fchb2 mk b2rg g bnnko kfgf tpmo kb nfdke okfgf mtnk2cfz nbpgz
brgmk fgf bngfkl bxzko pck2 tce ngb kfkf nl bngfko 1f nl vsn 2pgrf b1z
c1po b1 p2i2 nbzw ngmk br bngfkl bn yr1n1 zctko n1grbgn1gpz ngb 2yf pnbm2l
cfp2l mk tc ihyf f1b 2ip 1gzg "
QDeutNDt = "gkyzb p1f bn mn k1zbn gkbpz bnkfo 1ptf k1zbn b2 figyko gb2
fp1vjko b1z brgmk egcz cbghrkmo fkgo gnpe2o bg2o g1pz2o nt1g2o kfgf
bngfkrg mz2 tprg czk2 cigzc nb b2 bcg2krg mz2 kfgf b2 fczk2 fhg2 mk b2rg
brirg bnf vf fkgo mgnrg ikko fkgo vrko cvrko ecz kfgf tpmo cfz p2gl fb1
gbrgmk tgpe ckq kfgf gckrkmo ct2 ffkb nfdke nmo b2 eczk kfgf bngfkmo mk
kzb2o pvrk fb1 gngb tnk2o cfz nbpz brgmk kfgf bngfkl b1z fgxb2kl pbzw
pxzko pck2 tceko ngb kfkf nl bngfko bizko tn vrk nb 2t1f nl vsn gmgn 2pgrf
b1z c1pko pptn gb1z cbzw p2i2 gb1z cpko p2i2 nbzw ngb 212igf nfo gngb
2tgceo mk brgmk kfgf bngfkl bn yrb vgye tggq bcg2 tn crko tn 1n1ko gtn
zctko n1rbk tg1f ise nbnvko nbgfck gn1gpzk pxgg2k ngb 21b b2 1o kfgf
bngfkl n1gb mk ngb kryf kfgf b2 b1z k1b b2 1pg n1gb 1pgz b2 kgo f1c2 nye1g
mb1z xgl kfgf bngfkl 112 kpko 2tcge gt1k2 b2 mgn pnbm2l gckgo f1cktk 1c2
nkfgf bngfkl ngb 2t1f cg mn pnbmf b2f c2l c2l tcel gbp2l 1gzl gipgzl
gcfp2l dzkl b1z c1tzkl nptq krgi tcel gbp2l mpgl ghmz2f mk tce fkk2 cbzw
pxzko gkxkbl kfgf bngfkl p1o cke ihyf gchzgt rjgkf tn mq xgl kfgf bngfkl
n1pgz b2 kgo f1c2 nye1g mk 112 kpko t1f kfgf b2 f1pko gb2 fbzw b2 fko gmgn
b1z co gkrgi ckgo f1cktk tn mq czl kfgf b2 kgo f1c2 nye1g mce b2 bckl gb2
bpl mb1z xgl kfgf bngfkl nptq kbzkmgq kpkl gnptq kkjc nl tn fbepf b1z kfgf
bngfkl rg2q nl ngb 2zxi ngb 2rbu ngb 2drgc ngb 2trf cztkl te 1gb ngb 2ipge
b12 ztkl ngb 2ipge ck2 ztkl 1efg tceg bp2g 1gzg ipgzg gmgn b1z nztkl "
QDeutODt = "fo 1ptf mo fkgo gnp zk2 cizc nb brirg bnf vf f brmk tpe kzb2o
gb1z cpko p2i2 nk brmk kfgf "
QPhylADt = "figyko gb2 fp1vjko b b2rg brirg bnf vgb mgnrg ikk gngb tnk2pf
cfz nbz c1pko pptn gb1z c zctko n1grbk gt1f ise nnv xgmf kf bngfkmf gmgn c
"
QPhylBDt = "brgmk gpxgmf gnpe2pf bg2pf g1pz2pf nt1g2pf kfgf bngfkrg mz tpr
egcz cbghrkmpf fkgo fhf vrko cvrko ecz kfgf tpmpf cfz p2gl fb1 gbrgmk tgpe
ckq kfgf gckrmpf brgmk kfgf bngfkmf b1z fgxb2kmf p pxzko pck2 tceko ngb
b1z cpko p2i2 nbzw nb 212igf nfpf gngb 2tgcepf rgmk gn1gpzk pbg2k ngb 21b
b2 1o kfgf bngfkmf nb1g mk ngb kryf 2tcge gt1k2f b2 mgn pm2mf gcko f1ctk
1c2 nkf bngfkmf b1z c1tzkmf npt krgi tcemf gbp2mf mp2f mk bngfkmf nt1g2 b2
kgo f1c2 nye1g mce b2 bckmf gb2 bpmf ngb 2zxi gngb 2rbu gngb 2drgc gngb
2trf cztkmf te 1g gngb 2ipge "
QPhylGDt = "gkyzb pg1f bn mgn k1zbn gkbpz bnkfo 1ptf k1zbn b2 figyko b2
fp1vjko b1z brmk pxgmf fkgo tn f ecghrkmo fkgo fhf g1pz2o nt1g2o mk bn
kfgf bnfko tko b2o p2gl fb1 cfz p2gl fb1 gbrmk ke eczk czko fhf gnpe2o
bg2o b2 crkmo 2o tp cigzc ngb b2 bcg2krg mz2 kfgf b2 mk b2rg brirg vf bnf
mnrg ikko fkgo vrko cvrko fgf tpmo p2gl cfz fb1 gbrbrmk tgpe ckq bnfko mo
ct2 ffk gngb tnk2o cfz nbpgz brmk kfgf bnfkmf 2kl pbzw pxzkopck2 tceko nb
kfkf nl bnfko bizko b 2t1f nl vs gmgn 2pgrf b1z c1p pptn gb1z b1z cko p2i2
nbzw ngb 212igf nfo gngb nfkl bn yrb vgye tggq bcg2 tn crko gtn k gtg1f
ise nbnvko nbfck gn1pzk pxg2 bnfkl n1gb mk ngb kryf kfgf b2 b1z k1b b2 2 n
112 kpko 2tce gt1 1k2 mgn pnbm2l gcf bnfkl f1cktk nkfgf bnfkmf ngb 21f mgf
gcrl gc2l tce c2l 2l g1gzl gipzl gmgn cf1z c1tzkl mk 112 kfgf b2 fpko gb2
```

```
fo gb2 mgn b1z co gkrgi c k tn mq cl kfgf b2 c2 gye1fg mc bc pkl n fbepf
b1f bnfklf mb1z x xi ngb 2rbu ngb 2drc b 2trf cztkl ngb 2ipge ck2 ztkl
sefg tceg gbp2g 1gz gipzg gmgn b1z nztkl "
QPhylJDt = "gkyzb pg1f 1zbn bnkfpf 1ptf k1zbn 2 figyko gb2 fp1vjko 1z
brgmk egcz cbghrkmpf fkgo fhf gnpe2pf bg2pf g1pz2pf nt 2pf kfgf bngfkrg
mz2 tprg czk2 igzc gngb b2 bcg2krg mz2 kfgf b2 f czk2 fhgb2 mk b2rg brirg
bnf vgf mgnrg i kko fkgo vrko cvrko ecz kfgf tpmpf cfz p2gl fb1 gbrgmk
tgpe ckq kfgf gckrmpf ct2 ff bf nfdke nmpf b2 eczk kfgf mk kzb2pf pvrk fb1
ngb tnk2pf cfz nbpgz brgmk kfgf bngfkmf b1z fgx b2kmf pbzw pxzko pck2
tceko ngb kfkf nmf bngfko bizko tn vrk ngb 2t1f nmf vsn gmgn 2pgrf b1z
c1pko pptn gb1z cbzw p2i2 gb1z cpko p2i2 nbzw ngb 212igf nfpf gngb 2tgcepf
mk brgmk kfgf bngfkmf bn yrb vgye tgq bcg2 tn crko gtn 1n1ko gtn zctko
n1grbk tg1k ise nbnvko nbgfck gn1gpzk g2k ngb 21o kfgf bngfkmf nb1g mk nb
kryf kfgf b2 mgn b1z k1b b2 1pgz b2 kg xgmf kfgf bngfkmf 112 kpko 2tcge
gt1k2f b2 nbm2mf gcckgo f1c bngfkmf ngb 2t1f cgf mgn pnbmf b2f gcrmf gc2mf
tcemf g 2mf 1gzmf gipgzmf gmf gdzkmf b11tkmf nptq krgi tcemf gbp2mf mpgmf
ghm fk2f cbzw pbmf kfgf b1pf cke ihycbhzgt rjkf tn mq xgmf f mce b2 bck2
bpmf mb1z xfgf bngfkmf npmf gnptq kjc nm te 1g gngb 2ipg b12 ztkmf gge
ckkm gbp2zg gipg "
QPhylLDt = "o bizo tko p2i2 nbzw nb gtn 1n1ko gtn zckmf nb1g mk ngb krymf
112 kpko 2t2f gcrmf gc2mfrgi tcemf gbp2mfihyf gcbhzgt rjkgb2 bpmf mb1z
xgmf2q nmf ngb 2zxi gngb "
QPhylODt = "gkyzb pg1f bn rko cvrko ecz kfgf nk2pf cfz nbppgrf b1z c2 tn
crko gtn 1nmgn b1z k1b b2 c2 nkfgf bngp2mf m1g mce "
QPhylDt = "gkyzb p1f bn mn k1zpz b1pt k1zbn 2 figyko gb2 fp1vjz brmk fhf
gnpb2o2o nt1g2o kfgf bnfkrg mz2tmz2 czk2 cigzcnb b2 2 fczk2 fhb2 mkbrirg
bnf vf mnrg ikko vrko cvrko ecz kfgf tpmo p2glfgf gckrkmo2 ffb nfde nmo b2
ecz kfgf pvrk fb1 gnb tnk2o fz nbpzbrmfgxb2kl pbzwo pck2 tceko nb kfkf nl
bnfko bizko tn vrk 2t1fb1z cko pb1z cpko p2i2 nbzw nb 212igf nfo gnb 2tceo
mkbrmkvye tggqn crn 1n1ko gtn zctko gn1grb t1f ise nbnvko nbfck gn1pzk
pxg2 1o kb 1pgz b2 kgo f1c2 nye1g cmptgpckgo f gfkf2 kppngckgo f1cktk 1c2
nkfgf bnfkl nb 2t1fmn pnrlfp2l gdkz1tzkl "
BHSSESBDt = "gkyzb p1f bnmnk1zbn gkbpz bnfo 1pt k1zbn b2fiyko gb2fp1vjko
b1z brmk ecz cbhrkmo fkgo gnpe2o b2o g1pz2o nt12o kfgf bnfkrg mz2 tprg
czk2 cizc nb b2bc2krg mz2 kfgf b2fczk2 fhb2 mk b2rg brirg bnf vf fkgo mnrg
ikko vrko cvrko ecz kfgf tpmo cfz p2gl fb1 brmk tpe ckqkfgf gckrkmo ct2
ffgb nfdke nmo b2ecz kfgf mk kzb2o pvrk fb1 gnbtnk2o cfz nbpz s brmk kfgf
bnfkl b1z fgxb2kl pbzw pxzko pck2 tceko nb kfkfnl bnfko bizko tnvrk
nb2t1fnl vsn mn2pgrf b1z c1pko pptn gb1z cbzw p2i2 gb1z cpko p2i2 nbzw
nb212igf nfo gnb 2tceo mk brmk kfgf bnfkl bn yrb vye tgq bcg2 tncrko
gtn1n1ko gtnzctko n1rbk gt1f ise nbnvko nbfck gn1pzk pxg2g nb 21b b21okfgf
bnfkl n1gb mk nb kryf kfgf b2 b1zk1b b21pg n1gb s 1pgz b2kgo f1c2 nye1g
mb1z xgl kfgf bnfkl 112 kpko 2tce gt1k2 mnpnbm2l gkgo f1cktk 1c2 nkfgf
bnfkl nb 2t1f mnpnbmf b2f gcrlgc2l gtcelgbp2l g1gzl gipzl gmncfp2l gdzl
b1z c1tzkl nptq krgi tcel gbp2l mpgl ghmz2 mktce fkk2 cbzw pxzko gkxbl
kfgf bnfkl p1o cke ihyf gchzt rjgkf tnmq xgl kfgf bnfkl nt1g2 b2kgo f1c2 s
mce b2bckl gb2bpl mb1z xgl kfgf bnfkl nptq kbzkmq kpkl gnptq kkjc nl tn
fbepf b1zkfgf bnfkl r2q nl s nb 2zxi s gnb 2rbu s gnb 2drc s gnb2trf cztl
te 1gb s gnb 2ipe b12 ztl s gnb 22bgf ck2 ztl 1efg gtceg gbp2g 1gzg gipzg
gmn b1z nztl s "
LXXMTParallelDt = "gkyzb p1f bn mn k1zbn gkbpz bnfo 1pt k1zbn b2 fiyko gb2
fp1vjko b1z brmk ecz cbhrkmo fkgo gnpe2o b2o g1pz2o nt12o kfgf bnfkrg mz2
tprg czk2 cizc nb b2 bc2krg mz2 kfgf b2 fczk2 fhb2 mk b2rg brirg bnf vf
fkgo mnrg ikko vrko cvrko ecz kfgf tpmo cfz p2gl fb1 brmk tpe ckq kfgf
```

```
gckrkmo ct2 ffgb nfdke nmo b2 ecz kfgf mk kzb2o pvrk fb1 gnb tnk2o cfz
nbpz brmk kfgf bnfkl b1z fgxb2kl pbzw pxzko pck2 tceko nb kfkf nl bnfko
bizko tn vrk nb 2t1f nl vsn mn 2pgrf b1z c1pko pptn gb1z cbzw p2i2 gb1z
cpko p2i2 nbzw nb 212igf nfo gnb 2tceo mk brmk kfgf bnfkl bn yrb vye tgq
bcg2 tn crko gtn 1n1ko gtn zctko n1rbk gt1f ise nbnvko nbfck gn1pzk pxg2g
nb 21b b2 1o kfgf bnfkl n1gb mk nb kryf kfgf b2 b1z k1b b2 1pg n1gb 1pgz
b2 kgo f1c2 nye1g mb1z xgl kfgf bnfkl 112 kpko 2tce gt1k2 mn pnbm2l gkgo
f1cktk 1c2 nkfgf bnfkl nb 2t1f mn pnbmf b2f gcrl gc2l gtcel gbp2l g1gzl
gipzl gmn cfp2l gdzl b1z c1tzkl nptq krgi tcel gbp2l mpgl ghmz2 mk tce
fkk2 cbzw pxzko gkxbl kfgf bnfkl p1o cke ihyf gchzt rjgkf tn mq xgl kfgf
bnfkl nt1g2 b2 kgo f1c2 mce b2 bckl gb2 bpl mb1z xgl kfgf bnfkl nptq
kbzkmq kpkl gnptq kkjc nl tn fbepf b1z kfgf bnfkl r2q nl nb 2zxi gnb 2rbu
gnb 2drc gnb 2trf cztl te 1gb gnb 2ipe b12 ztl gnb 22bgf ck2 ztl 1efg
gtceg gbp2g 1gzg gipzg gmn b1z nztl "
SamPent = "b1e2 fvsdf gkyzb p1f bn mn k1zbn gkbpz bnfo 1pt k1zbn b2 fiyko
gb2 fp1vjko b1z brmk ecz cbhrkmo fkgo gnpe2o b2o g1pz2o nt12o kfgf bnfkrg
mz2 tprg czk2 cizc nb b2 bc2krg mz2 kfgf b2 fczk2 fhb2 mk b2rg brirg bnf
vf fkgo mnrg ikko vrko cvrko ecz kfgf tpmo cfz p2gl fb1 brmk tpe ckq kfgf
gckrkmo ct2 ffgb nfdke nmo b2 ecz kfgf mk kzb2o pvrk fb1 gnb tnk2o cfz
nbpz brmk kfgf bnfkl b1z fgxb2kl pbzw pxzko pck2 tceko nb kfkf nl bnfko
bizko tn vrk nb 2t1f nl vsn mn 2pgrf b1z c1pko pptn gb1z cbzw p2i2 gb1z
cpko p2i2 nbzw nb 212igf nfo gnb 2tceo mk brmk kfgf bnfkl bn yrb vye tgq
bcg2 tn crko gtn 1n1ko gtn zctko n1rbk gt1f ise nbnvko nbfck gn1pzk pxg2g
nb 21b b2 1o kfgf bnfkl n1gb mk nb kryf kfgf b2 b1z k1b b2 1pg n1gb 1pgz
b2 kgo f1c2 nye1g mb1z xgl kfgf bnfkl 112 kpko 2tce gt1k2 mn pnbm2l gkgo
f1cktk 1c2 nkfgf bnfkl nb 2t1f mn pnbmf b2f gcrl gc2l gtcel gbp2l g1gzl
gipzl gmn cfp2l gdzl b1z c1tzkl nptq krgi tcel gbp2l mpgl ghmz2 mk tce
fkk2 cbzw pxzko gkxbl kfgf bnfkl p1o cke ihyf gchzt rjgkf tn mq xgl kfgf
bnfkl nt1g2 b2 kgo f1c2"

fragments = list(BHSSESBDt, QPhylDta, QPhylRDt, QDeutJDt, QDeutNDt,
QPhylADt, QPhylBDt, QPhylGDt,
                 QPhylJDt, QPhylLDt, QPhylODt, QPhylDt,
                 LXXMTParallelDt, SamPent)
fragment_names = c("BHSSESBDt", "QPhylDta", "QPhylRDt", "QDeutJDt",
"QDeutNDt", "QPhylADt", "QPhylBDt", "QPhylGDt",
                  "QPhylJDt", "QPhylLDt", "QPhylODt", "QPhylDt",
                  "LXXMTParallelDt","SamPent")

percent_match_threshold = 0.44 #ave = 23 sd = 21


beginning_time = proc.time()
match_all_sequences(fragments, fragment_names)
total_time = proc.time() - beginning_time
total_time
print(paste(matches/total_comparisons, "of the comparisons
matched.",matches," did. ", total_comparisons-matches," did not.")
```

# BIBLIOGRAPHY

Brug, J.F. *Textual Criticism of the Old Testament.* LULU Press, 2014.

Evans, Craig A., and Emanuel Tov. *Exploring the Origins of the Bible : Canon Formation in Historical, Literary, and Theological Perspective.* Acadia Studies in Bible and Theology. Grand Rapids, Mich.: Baker Academic, 2008.

Fox, Michael V, *Proverbs : an eclectic edition with introduction and textual commentary*. Atlanta, Georgia:SBL Press, 2015.

Gurry, Peter J. *A Critical Examination of the Coherence-Based Genealogical Method in New Testament Textual Criticism.* New Testament Tools, Studies, and Documents,. Leiden: Brill, 2017.

Kraft, Robert A., Emanuel Tov, John R. Abercrombie, and Computer Assisted Tools for Septuagint Studies (Project). *Computer Assisted Tools for Septuagint Studies (Catss).* Septuagint and Cognate Studies Series. Atlanta, Ga.: Scholars Press, 1986.

Longacre, Drew. "A Contextualized Approach to the Hebrew Dead Sea Scrolls Containing Exodus." Ph.D., University of Birmingham, 2015.

Schiffman, Lawrence H., Emanuel Tov, James C. VanderKam, and Galen Marquis. *The Dead Sea Scrolls Fifty Years after Their Discovery : Proceedings of the Jerusalem Congress, July 20-25, 1997.* Jerusalem: Israel Exploration Society in cooperation with The Shrine of the Book, Israel Museum, 2000.

Schenker, Adrian, Jan de Waard, P. B. Dirksen, Yohanan Goldman, Rolf Schäfer, Magne Sæbø, David Marcus, and Carmel McCarthy. *Biblia Hebraica Quinta.: Deuteronomy.; Volume 18: General Introduction and Megilloth.; Volume 20: Ezra and Nehemiah.* Vol. 5. Stuttgart: Deutsche Bibelgesellschaft., n.d.

Talmon, Shemaryahu, Michael A. Fishbane, Emanuel Tov, and Weston W. Fields. *Sha'arei Talmon : Studies in the Bible, Qumran, and the Ancient near East Presented to Shemaryahu Talmon.* Winona Lake, Ind.: Eisenbrauns, 1992.

Tov, Emanuel. *Scribal Practices and Approaches Reflected in the Texts Found in the Judean Desert.* Studies on the Texts of the Desert of Judah,. Leiden ; Boston: Brill, 2004.

———. *Scribal Practices and Approaches Reflected in the Texts Found in the Judean Desert.* Studies on the Texts of the Desert of Judah. Atlanta: Society of Biblical Literature, 2009.

———. *Textual Criticism of the Hebrew Bible.* Third edition, revised and expanded. ed.

———. *The Parallel Aligned Hebrew-Aramaic and Greek Texts of Jewish Scripture*. Bellingham, WA: Lexham Press, 2003.

———. *The Text-Critical Use of the Septuagint in Biblical Research.* Jerusalem Biblical Studies. 2nd ed. Jerusalem: Simor, 1997.

Tov, Emanuel, and Martin G. Abegg. *The Texts from the Judaean Desert : Indices and an Introduction to the Discoveries in the Judaean Desert Series.* Discoveries in the Judaean Desert. Oxford: Clarendon Press, 2002.

Tov, Emanuel, Noel B. Reynolds, Brigham Young University., and Neal A. Maxwell Institute for Religious Scholarship. *The Dead Sea Scrolls Electronic Library*. Provo, Utah, Leiden: Bringham Young University ; Koninklijke Brill NV, 2006.

Wonneberger, Reinhard. *Understanding BHS: A Manual for the Users of Biblia Hebraica Stuttgartensia*. Vol. 8. 2nd rev. ed. Subsidia Biblica. Roma: Pontificio Istituto Biblico, 1990.