public class NumberUtils { public static final Long LONG\_ZERO = Long . valueOf ( 0L ) ;

public static final Long LONG\_ONE = Long . valueOf ( 1L ) ;

public static final Long LONG\_MINUS\_ONE = Long . valueOf ( ( - 1L ) ) ;

public static final Integer INTEGER\_ZERO = Integer . valueOf ( 0 ) ;

public static final Integer INTEGER\_ONE = Integer . valueOf ( 1 ) ;

public static final Integer INTEGER\_MINUS\_ONE = Integer . valueOf ( ( - 1 ) ) ;

public static final Short SHORT\_ZERO = Short . valueOf ( ( ( short ) ( 0 ) ) ) ;

public static final Short SHORT\_ONE = Short . valueOf ( ( ( short ) ( 1 ) ) ) ;

public static final Short SHORT\_MINUS\_ONE = Short . valueOf ( ( ( short ) ( - 1 ) ) ) ;

public static final Byte BYTE\_ZERO = Byte . valueOf ( ( ( byte ) ( 0 ) ) ) ;

public static final Byte BYTE\_ONE = Byte . valueOf ( ( ( byte ) ( 1 ) ) ) ;

public static final Byte BYTE\_MINUS\_ONE = Byte . valueOf ( ( ( byte ) ( - 1 ) ) ) ;

public static final Double DOUBLE\_ZERO = Double . valueOf ( 0.0 ) ;

public static final Double DOUBLE\_ONE = Double . valueOf ( 1.0 ) ;

public static final Double DOUBLE\_MINUS\_ONE = Double . valueOf ( ( - 1.0 ) ) ;

public static final Float FLOAT\_ZERO = Float . valueOf ( 0.0F ) ;

public static final Float FLOAT\_ONE = Float . valueOf ( 1.0F ) ;

public static final Float FLOAT\_MINUS\_ONE = Float . valueOf ( ( - 1.0F ) ) ;

public NumberUtils ( ) { }

public static int toInt ( final String str ) { }

public static int toInt ( final String str , final int defaultValue ) { }

public static long toLong ( final String str ) { }

public static long toLong ( final String str , final long defaultValue ) { }

public static float toFloat ( final String str ) { }

public static float toFloat ( final String str , final float defaultValue ) { }

public static double toDouble ( final String str ) { }

public static double toDouble ( final String str , final double defaultValue ) { }

public static byte toByte ( final String str ) { }

public static byte toByte ( final String str , final byte defaultValue ) { }

public static short toShort ( final String str ) { }

public static short toShort ( final String str , final short defaultValue ) { }

public static Number createNumber ( final String str ) throws NumberFormatException { } private static boolean isAllZeros ( final String str ) { }

public static Float createFloat ( final String str ) { }

public static Double createDouble ( final String str ) { }

public static Integer createInteger ( final String str ) { }

public static Long createLong ( final String str ) { }

public static BigInteger createBigInteger ( final String str ) { if ( str == null ) { return null ; } int pos = 0 ; int radix = 10 ; boolean negate = false ; if ( str . startsWith ( "-" ) ) { negate = true ; pos = 1 ; } <START\_BUG> if ( ( str . startsWith ( "0x" , pos ) ) || ( str . startsWith ( "0x" , pos ) ) ) { <END\_BUG> radix = 16 ; pos += 2 ; } else if ( str . startsWith ( "#" , pos ) ) { radix = 16 ; pos ++ ; } else if ( ( str . startsWith ( "0" , pos ) ) && ( ( str . length ( ) ) > ( pos + 1 ) ) ) { radix = 8 ; pos ++ ; } final BigInteger value = new BigInteger ( str . substring ( pos ) , radix ) ; return negate ? value . negate ( ) : value ; }

public static BigDecimal createBigDecimal ( final String str ) { }

public static long min ( final long ... array ) { }

public static int min ( final int ... array ) { }

public static short min ( final short ... array ) { }

public static byte min ( final byte ... array ) { }

public static double min ( final double ... array ) { } public static float min ( final float ... array ) { } public static long max ( final long ... array ) { } public static int max ( final int ... array ) { } public static short max ( final short ... array ) { } public static byte max ( final byte ... array ) { } public static double max ( final double ... array ) { } public static float max ( final float ... array ) { } private static void validateArray ( final Object array ) { } public static long min ( long a , final long b , final long c ) { } public static int min ( int a , final int b , final int c ) { } public static short min ( short a , final short b , final short c ) { } public static byte min ( byte a , final byte b , final byte c ) { } public static double min ( final double a , final double b , final double c ) { } public static float min ( final float a , final float b , final float c ) { } public static long max ( long a , final long b , final long c ) { }

public class Main {

private static final int nTrialsPerPaper = 12 ;

private Indexer < String > wordIndexer ;

private Indexer < PaperAbstract > paperIndexer ;

private Terms terms ;

private KNNSimilarityCache knnSimilarityCache ;

private KNNGraphDistanceCache knnGraphDistanceCache ;

private static MetadataLogger metadataLogger ;

private static Random randGen ;

public List < TrainingPaper > trainingSet ;

public List < PredictionPaper > testingSet ;

private Baseline baseline ; private CommonNeighbors cn ;

private DTRandomWalkPredictor dtRWPredictor ;

private KNN knn ; private KNNWithCitation knnc ;

private LSI lsi ;

public Main ( long randomSeed ) { }

public void load\_data ( String filename , double trainPercent ) { }

private void splitByTrainPercent ( double trainPercent , List < PaperAbstract > documents ) { }

public void splitHeldoutWords ( double testWordPercent ) { }

public static Random getRandomGenerator ( ) { }

public static MetadataLogger getMetadataLogger ( ) { }

public double [ ] evaluate ( PredictionPaper testingPaper , Integer [ ] prediction , int size , int k ) { }

public static void printResults ( double [ ] results ) { }

public static void printResults ( File output , double [ ] results ) { }

public void runClusteringMethods ( File outputDir , int [ ] ks ) { int size = ( trainingSet . size ( ) ) + ( testingSet . size ( ) ) ;

if ( Main . testIsEnabled ( "baseline" ) ) { baseline = new Baseline ( trainingSet , terms ) ;

runClusteringMethod ( testingSet , baseline , outputDir , ks , size ) ; }

if ( Main . testIsEnabled ( "dtrw" ) ) { int rwLength = Integer . getInteger ( "plusone.dtrw.walkLength" , 4 ) ; boolean stoch = Boolean . getBoolean ( "plusone.dtrw.stochastic" ) ;

int nSampleWalks = Integer . getInteger ( "plusone.dtrw.nSampleWalks" ) ; System . out . println ( ( "Random<seq2seq4repair\_space>walk<seq2seq4repair\_space>length:<seq2seq4repair\_space>" + rwLength ) ) ; if ( stoch ) System . out . println ( ( ( "Stochastic<seq2seq4repair\_space>random<seq2seq4repair\_space>walk:<seq2seq4repair\_space>" + nSampleWalks ) + "<seq2seq4repair\_space>samples." ) ) ;

boolean finalIdf = Boolean . getBoolean ( "plusone.dtrw.finalIdf" ) ;

boolean ndiw = Boolean . getBoolean ( "plusone.dtrw.normalizeDocsInWord" ) ; Boolean nwid = Boolean . getBoolean ( "plusone.dtrw.normalizeWordsInDoc" ) ; dtRWPredictor = new DTRandomWalkPredictor ( trainingSet , terms , rwLength , stoch , nSampleWalks , finalIdf , nwid , ndiw ) ; runClusteringMethod ( testingSet , dtRWPredictor , outputDir , ks , size ) ; } int [ ] closest\_k = Main . parseIntList ( System . getProperty ( "plusone.closestKValues" , "1,3,5,10,25,50,100,250,500,1000,10000,100000" ) ) ; KNNSimilarityCacheLocalSVDish KNNSVDcache = null ; LocalSVDish localSVD ; KNNLocalSVDish knnSVD ; if ( Main . testIsEnabled ( "svdishknn" ) ) { int [ ] TODOpar = new int [ ] { 50 , 50 , 50 } ; localSVD = new LocalSVDish ( 3 , TODOpar , TODOpar , TODOpar , TODOpar , TODOpar , trainingSet , terms . size ( ) ) ; KNNSVDcache = new KNNSimilarityCacheLocalSVDish ( trainingSet , testingSet , localSVD ) ; } for ( int ck = 0 ; ck < ( closest\_k . length ) ; ck ++ ) { if ( Main . testIsEnabled ( "knn" ) ) { knn = new KNN ( closest\_k [ ck ] , trainingSet , paperIndexer , terms , knnSimilarityCache ) ; runClusteringMethod ( testingSet , knn , outputDir , ks , size ) ; } <START\_BUG> if ( Main . testIsEnabled ( "KNNLocalSVDish" ) ) { <END\_BUG> knnSVD = new KNNLocalSVDish ( closest\_k [ ck ] , trainingSet , paperIndexer , terms , KNNSVDcache ) ; runClusteringMethod ( testingSet , knnSVD , outputDir , ks , size ) ; } if ( Main . testIsEnabled ( "knnc" ) ) { knnc = new KNNWithCitation ( closest\_k [ ck ] , trainingSet , paperIndexer , knnSimilarityCache , knnGraphDistanceCache , terms ) ; runClusteringMethod ( testingSet , knnc , outputDir , ks , size ) ; } } int [ ] dimensions = Main . parseIntList ( System . getProperty ( "plusone.svdDimensions" , "1,5,10,20" ) ) ; for ( int dk = 0 ; dk < ( dimensions . length ) ; dk ++ ) { if ( Main . testIsEnabled ( "lsi" ) ) { lsi = new LSI ( dimensions [ dk ] , trainingSet , terms ) ; runClusteringMethod ( testingSet , lsi , outputDir , ks , size ) ; } } }

public void runClusteringMethod ( List < PredictionPaper > testingSet , ClusteringTest test , File outputDir , int [ ] ks , int size ) { } static double [ ] parseDoubleList ( String s ) { } static int [ ] parseIntList ( String s ) { } static Boolean testIsEnabled ( String testName ) { } public static void main ( String [ ] args ) { } No newline ; at end ; of file ; }