Evaluation of Clinical Concept Embeddings Trained on General Medical Text by Field of Medicine

John-Jose Nunez
Depts. of Psychiatry and Computer Science, UBC
jjnunez@cs.ubc.ca

1 Introduction

1.1 Background

The application of natural language processing and machine learning to medicine presents an exciting opportunity for tasks requiring prediction and classification, such as predicting the risk of suicide after a patient is discharged from hospital (McCoy et al., 2016). A common approach is to convert the unstructured text produced by clinical interactions into low-dimension vector representations which can then be fed into these algorithms. These vectorizations are produced by training models on large unlabelled corpora. For example, the popular *word2vec* system (Mikolov et al., 2013) initially trained embeddings using a skip-gram model, producing a target word's vector based on what words surround it. It was initially trained on a Google News corpus containing around six billion tokens. Due to considerable differences between the language of medical text and general English writing, prior work has trained medical embeddings using specific medical sources.

Generally, these approaches have trained embeddings to represent medical concepts according to their 'clinical unique identifiers' (CUIs) in the Unified Library Management System (ULMS) (Bodenreider, 2004). Words in a text can then be mapped to these CUIs (Yu and Cai, 2013). Examples include De Vine et al (2014) who trained embeddings using journal abstracts from MEDLINE as well as with clinical patient records. They evaluated these embeddings by comparing vector cosine similarity against human-judged similarity. Minarro-Gimenez et al (2014) trained embeddings using medical manuals, articles, and Wikipedia articles, judging quality by their ability to predict drug-disease relations noted in the National Drug File - Reference Terminology (NDF-RT) ontology. Choi et al (Choi et al., 2016) built on this work by learning two sets of embeddings, from health insurance claims and clinical narratives. They evaluated their embeddings by their ability to predict known relations including those in the NDF-RT, disease hierarchies, and medical concept type. In their yet unpublished work, Beam et al (Beam et al., 2018) learn embeddings on the largest dataset yet, combining health insurance claims, clinical narratives and full journal texts. They developed a new system termed "cui2vec", training CUI embeddings based on the occurrences of other identifiers within a certain window length. They use an assortment of aforementioned known relations to compare the quality of these embeddings.

All of the above examples were both trained and evaluated on general medical data, from all fields of medicine. It is unclear how these models perform in specific fields of medicine. For example, we can consider the medical speciality of psychiatry, the field of medicine concerned with mental illness such as depression or schizophrenia. Prior work has shown that psychiatric symptoms are often described in a long, varied, and subjective manner (Forbush et al., 2013 3 18) which may present a particular challenge for training these embeddings and NLP tasks generally.

Prior work has explored whether domain adaptation (DA), techniques to adapt data from other domains to work on a target, can improve performance when applied to this sub-domain of psychiatry. Lee et al (Lee et al., 2018) used these techniques to improve the task of de-identifying psychiatric notes. Zhang et al (Zhang et al., 2018) then applied DA to word embeddings trained from general language and medical sources, showing some improvements when targeting a psychiatric dataset.

1.2 Contribution

In this work, we seek to start understanding how NLP performance may vary when applied to difference fields of medicine. Specifically, we compare the quality of embeddings trained on general medical data by their related field of medicine, using a variety of metrics previously described in the literature. As NLP is applied to medicine, field-specific applications will become increas-

ingly popular. If this work finds little difference between fields of medicine, future researchers will be assured that embeddings trained from general medical text will be sufficient. Conversely, if differences are found for specific fields, future work may want to address this shortfall by using techniques like DA, or even creating embeddings specifically trained for specific fields.

Additionally, this project also contributes to the evaluation of medical embeddings. Existing methods generally evaluate whether the embedding vectors for a given concept are similar to other embeddings given a known relation. This work is the first to use a concept's field of medicine as such a relation, which we believe may be more clinically relevant than some used previously.

Lastly, by comparing multiple sets of embeddings using different evaluation metrics, this work seeks to also evaluate the relative performance of embeddings trained with different methods.

2 Methods

2.1 Obtaining Embeddings

Table 2.1 contains details of the embeddings compared in this project, all of which are based on *word2vec*. We obtained DeVine200 (De Vine et al., 2014), ChoiClaims300, and ChoiClinical300 (Choi et al., 2016) all from the later's Github. We downloaded BeamCui2Vec500 (Beam et al., 2018) from this site.

2.2 Evaluation Methods

Determining a Clinical Concept's Field of Medicine A clinical concept's corresponding field of medicine is not necessarily obvious. In order to have an objective and unambiguous classification, we utilized the ninth revision of the International Statistical Classification of Diseases and Related Health Problems (ICD-9) (Slee, 1978). This is a widely used system of classifying medical diseases and disorders, dividing them into seventeen chapters representing medical systems/categories such as mental disorders, or disease of the respiratory system. While the 10th version is available, we chose this version based on prior work using it, and the pending release of the 11th version. We will refer to these ICD9 systems as medical systems throughout this work.

We consider CUIs representing diseases/conditions which have known ICD9 equivalents according to a CUI-to-ICD9 dictionary available from the UMLS. All ICD9 codes are within ranges that specify the medical system. We also consider pharmacological substances related to a medical system if they treat or prevent a disease with an ICD9 code within a particular ICD9 system. We determine this by using the NDF-RT dictionary, which maps CUIs of substances to the CUIs of conditions they treat or prevent, and then convert these CUIs to the ICD9 systems as before.

Medical Relatedness Property (MRP)

This metric from (Choi et al., 2016) is based on quantifying whether concepts with known relations are located near each other. They use, separately, known relationships between drugs and the diseases they treat or prevent, and the relations between diseases that are grouped together in the CCS hierarchical groupings, a classification from the Agency for Healthcare Research and Quality. The scoring utilizes Discounted Cumulative Gain, which attributes a diminishing score the further away a known relationship is found if within k neighbours.

In our implantation, we calculate the MRP based on the 'course' groupings from CCS drug hierarchies. Scores are calculated for CUIs that represent diseases with a known ICD9 code. The mean MRP is then calculated for all CUIs within a given ICD9 system. The implementation was adapted from Python 2.7 code available from the author's Github.

Medical Conceptual Similarity Property (MCSP) The other metric used by Choi et al's work evaluates whether embeddings known to be of a particular set are clustered together. They use conceptual sets from the UMLS such as 'pharmacologic substance' or 'disease or syndrome'. Discounted Cumulative Gain is again used, based on whether a CUI has other CUIs of its set within k neighbours.

We reimplement this method, but instead of using the UMLS conceptual sets, we create sets from the ICD9 systems, again giving a score to neighbours that are conditions or substances from the same medical system. Again, this was adapted from code from Choi et al's Github.

Correlation with UMNSRS Similarity (SimCor) (Yu et al., 2017) investigate whether the cosine similarity of embedding vectors are correlated with human judgements of similarity. The UMNSRS-Similarity dataset (Pakhomov, 2018) contains around 500 similarity ratings between medical concepts as rated by eight medical residents. Yu et all then compute a Spearman rank correlation between the cosine similarities and the UMNSRS-Similarity ratings.

We repeated the above, calculating a medical system's mean correlation based on the Spearman rank correlation of all pairing that contain at least one disease with an ICD9 code in its system, or a drug that treats or prevents a disease in the system. This was implemented from scratch in Python, and we used the UMN SRS modified similarity dataset.

Significance against Bootstrap Distribution (Bootstrap) Beam et al (2018) also evaluate how well known

Name	Dimension	Number	Number of Training Data	Type of Training Data
DeVine200	200	52,102	17k + 348k	clinical narratives, journal abstracts
ChoiClaims300	300	14,852	4m	health insurance claims
ChoiClinical300	300	22,705	20m	clinical narratives
BeamCui2Vec500	500	109,053	60m + 20m + 1.7m	claims, narratives, full journal texts

Table 1: Characteristics of the embeddings compared, including the name referred, the embedding dimensions, the number of embeddings in the dataset, and the type of data used to train them.

relationships between concepts are borne out in embedding vector similarity. For a given type of relation, they generate a bootstrap distribution by randomly calculating cosine similarities of embedding vectors of the same class (eg. a random drug and disease when evaluating may-treat relations). For a given known relation, they consider that the embeddings produced an accurate prediction if their cosine similarity is within the top 5%, the equivalent of p¡0.05 for a one-sided t-test.

Our implementation considers the may-treat or may-prevent known relationships from the NDF-RT dataset. We calculate the percentage of known relations for drug-disease pair within each medical system. Beam et al have not yet made their code publicly available, so this code was written in entirety in Python.

System Vector Prediction MeanVec(SysVec) We implement a new method to evaluate embeddings. A representative vector is calculated for each medical system by calculating the mean of normalized embeddings of a system's conditions. We then calculate the percentage of drugs known to treat or prevent a disease in each system whose vectors are most similar (by cosine similarity) to the relevant system vector. For example, we would expect the CUI for 'fluoxetine', an anti-depressant, to be most similar to the Mental Disorders centroid.

Some drugs treat or prevent diseases in n multiple system. For a medical system, such a drug is considered being accurately predicted if the system's centroid is amongst the n most similar centroids. We again wrote this in Python.

Analysis Our work seeks to determine if embeddings for one medical system are worse or better than others, or if those from a set of embeddings are worse or better. To do this, we must consider the scores generated using five different metrics, four different embeddings, across the seventeen medical systems. However, the scores from five metrics are not obviously convertible to a common score.

For now, we will assume our results are normally distributed; this may not be unreasonable as the processes underlying the quality of embeddings - the use of words representing clinical concepts in texts - stem from a natural process (human writing word choice).

To compare the relative scores for each medical systems, for each evaluation method we calculate the mean score for the system's embeddings in each embedding set. We then conducted a paired two-tailed t-test for four pairs. Each pair contains a system's mean score vs the mean of all scores, for one embedding set. We then compare the relative difference between these means, express it as a percentage, and report whether this was significant at $p < 0.05$.

We repeat the same steps to compare the sets of embeddings. This time, we calculate the mean scores from each medical system for each evaluation method, and conduct the paired two-tailed t-tests on seventeen pairs for each medical system. The pairs are the mean score from all the embedding sets, against the score from the given embedding set. Again, we calculate the set's score relative to mean of all sets, and the significance as above.

After observing that the embeddings from Beam et al outperform the other embedding sets, we then use only these embeddings to evaluate the medical systems by the MCSP method. We chose this method as it resulted in the highest number of significant differences, and is able to calculate scores for all systems.

Choosing one embedding set allows more embeddings to be compared, as previous scores were only calculated on embeddings shared by all embedding sets. As well, choosing one evaluation method allows statistical analysis to be performed on the individual embedding scores instead of means. As such, for each medical system we then perform a one-sided two-tailed t-test between its embeddings and the scores from all relevant embeddings. We then repeated the same but only for the embeddings representing CUIs that were overlapping between the four embedding sets. This investigates the effect of restricting which CUIs are included.

3 Results

3.1 Differences Between Medical Systems

Comparing the embeddings from medical systems across the five evaluation methods reveals that some systems have scores significantly above the mean across multiple methods (Table 3.1). Embeddings related to cancers and musculoskeletal system are significantly above the

mean in two methods, while those of mental disorders, the nervous system, and the cardiovascular system are significantly above in three methods. No systems have scores significantly below the mean more than once. Embeddings related to diseases of pregnancy, the perinatal period, and skin disorders appear to perform worst.

3.2 Differences Between Sets of Embeddings

Evaluating the sets of embeddings (Table 3.1) shows some stark differences. The *cui2vec* embeddings from Beam et al are above the mean across all evaluation methods. Those from DeVine et al, and those from Choi et al based on the clinical narratives, do significantly worse. The remaining embeddings, those based on health insurance claims from Choi et al, are more middling.

3.3 Differences when Constraining Number of Embeddings

In Table 3.3 we focus on MCSP scores from the best performing set of embeddings, those from Beam et al. We observe results that are somewhat similar to the main analysis. Embeddings related to mental disorders and the nervous system again perform well. Cardiovascular embeddings are better only when the more restrictive set of overlapping embeddings are considered. We note large differences for systems that had few overlapping embeddings but many only in the Beam et al embeddings, such as those related to pregnancy and injury and poisonings. While these results descriptively seem to match for many systems, metrics indicate poor correlation. Assuming normality, Pearson's coefficient is only 0.16, while not assuming normality and using Spearman's we get only 0.01.

4 Discussion

4.1 Differences Between Medical Systems

In this project, we seek to determine whether embeddings for clinical concepts (CUIs) learned from general medical text work similarly well for the various fields of medicine. We investigate this by using different metrics of embedding quality and sets of embeddings, and use ICD9 systems to determine an embedding's relevant medical fields. Based on the methods used so far, our results suggest that embeddings from certain medical systems perform better than others- namely, those of mental disorders, the nervous system, and possibly those of the cardiovascular and musculoskeletal systems. It is less clear if any systems perform particularly poorly, though some of the systems had few or no relevant embeddings for some metrics.

Why some of these systems seem to perform better is unclear. Both mental and neurological disorders relate to the brain, but their respective fields of medicine are otherwise dissimilar across frequency, language, and specificity of diseases. If we take the numbers of compared embeddings as a proxy for how common these medical system's conditions are, the well performing systems are neither particularly common or uncommon. As the frequency a concept occurs in training data may impact the quality of its embedding, this may be something to examine going forwards.

Strengths of the project include an objective way of relating a clinical concept to a field of medicine. As well, the combination of multiple evaluation metrics and sets of embeddings encompasses most of the prior work in this topic. However, these complexities also lead to weaknesses. First and foremost, the combination of different metrics, embedding sets, and medical systems creates a daunting task for statistical analysis, and this could likely be improved by considering advanced techniques such as non-parametric equivalents to ANOVA and ensuing post-hoc analysis.

Additionally, we see that there is a difference in the results when fewer or greater numbers of embeddings are considered for a given system. The initial analysis only considered embeddings that were common to all embedding sets, significantly restricting the numbers of embeddings considered compared to the relevant concepts within the most extensive *cui2vec* embeddings from Beam et al. One possible cause of the difference may be simply sampling too few embeddings; for some systems, only a few dozen examples had embeddings in all four embedding sets. However, the score difference may also be the result of selecting less common medical terms; the set of overlapping embeddings likely represent more common clinical concepts if all four training methods include them.

This last point is important to consider when assessing the project's results. The differences we find between medical systems could be due to how many rare clinical concepts a system has; these rare terms may have had fewer chances to be trained well due to seldomly occurring in the training data. This may or may not be a confounder. A future researcher using the embeddings in a particular field of medicine may be using embeddings from a variety of concepts in her field, including rare ones, in which case worse performance due to a field containing rare terms would be desired. Conversely, applications may only consider common terms in a field, in which case these rare terms would indeed be confounding. This limitation can be addressed, as will be discussed below. However, our main analysis does only consider CUIs in all four embedding sets, which may itself limit the rarity of which embeddings are compared.

ICD9 Medical System	MRP (%)	MCSP (%)	SimCor (%)	Bootstrap (%)	SysVec (%)
Infections	-21	+41	**-34**	**+35**	+12
Cancers	**+19**	+2	**+51**	+38	-4
Endocrine	+3	**-12**	**+20**	+23	-10
Blood diseases	-7	**-29**	-2	-6	+10
Mental disorders	**+23**	**+67**	-45	+25	**+33**
Nervous system	**+61**	**+64**	-3	+17	**+41**
Cardiovascular	**+12**	**+64**	+19	+22	**+38**
Respiratory	-13	+7	**-24**	**+40**	-7
Digestive	+24	-7	**-42**	-7	-39
Genitourinary	+29	+8	-15	+7	**+18**
Pregnancy	-22	**-56**			
Skin	**-28**	-20	-31	+18	+8
Musculoskeletal	+5	+13	**+47**	+6	**+31**
Congenital	-15	-13		+11	**+52**
Perinatal	-38	**-57**			
Ill-defined	-18	-28	-24	-18	+5
Injury and poisoning	-14	**-44**	**+84**	-12	+13

Table 2: Percentage difference of a medical system's embeddings vs the mean score for all considered embeddings. Significant (paired t-test p <0.05) scores are in orange (above mean) and blue (below). See Methods section for method abbreviations. Blank values represent no scores could be calculated for a system with that method.

Embedding Set	MRP (%)	MCSP (%)	SimCor (%)	Bootsrap (%)	SysVec (%)
DeVine200	**-35**	**-12**	+7	**-46**	-4
ChoiClaims300	**+10**	+4	-6	-4	2
ChoiClinical300	**-14**	**-16**	-5	-2	-1
BeamCui2Vec500	**+38**	**+24**	+4	**+52**	+3

Table 3: Percentage difference of an embedding set's mean scores vs those of all embedding sets. Significant (paired t-test p <0.05) scores above are shown in orange, below blue. See Methods section for embedding set and evaluation method abbreviations.

4.2 Differences Between Sets of Embeddings

Another goal of the project was to assess the quality of each set of embeddings. We see that Beam et al's *cui2vec* embeddings clearly come out ahead. This is expected, as these embeddings are the most recent, are trained on the largest amount of data, and are trained on three different types of data. They also contain the largest number of embeddings. While we are only comparing four sets, our results may suggest that health insurance claim data is particularly useful to train embeddings, given that the two sets of embeddings containing this data performed better.

These results also suggest that our use of ICD9 systems as a relationship to judge embedding vectors - such as whether embeddings from a given system are near each other - are a contribution to the field.

4.3 Comparing Evaluation Methods

Our work also allows us to compare embedding evaluation methods. It is difficult to judge them concretely, as we do not have a gold standard to tell us what results a method should produce.

Yu et al's method of comparing correlation with similarity as judged by resident physicians was hampered by having a small number of judged similarities (around 500) which led to few comparisons per system in our deployment. Our new method utilizing 'system vectors' observed similar differences between medical systems as found by other methods. However, it found very small differences between sets of embeddings. Neither of these two methods found the *cui2vec* embeddings to be significantly better than the others. If we assume this difference should be found, this may suggest these two methods are inferior. Our implementation of Yu et al's method could likely be slightly improved; some of the human similarity comparisons are not used as the CUIs chosen are not found within our embeddings, so could be changed to very related ones that are, eg Diabetes to Diabetes Mellitus.

The two methods from Choi et al produced similar results. This could likely be expected, as they are simi-

	All Relevent Embeddings			Overlapping Embeddings			
ICD9 System	MCSP	Examples	Difference (%)	MCSP	Examples	Difference (%)	Different?
All	8.07	16351	0	5.60	2884	0	Yes
Infections	7.72	2261	**-4**	6.84	334	**+22**	Yes
Cancers	9.00	1194	**+12**	4.47	116	**-20**	Yes
Endocrine	5.64	545	**-30**	3.98	193	**-29**	Yes
Blood Diseases	4.36	199	**-46**	3.67	81	**-34**	Yes
Mental Disorders	9.34	662	**+16**	7.67	165	**+37**	Yes
Nervous	8.44	1787	**+5**	7.27	434	**+30**	Yes
Cardiovascular	8.12	869	+1	7.74	307	**+38**	No
Respiratory	5.85	405	**-27**	4.64	132	**-17**	Yes
Digestive	7.93	852	-2	4.62	210	**-18**	Yes
Genitourinary	6.82	606	**-15**	5.75	210	+3	Yes
Pregnancy	10.27	1325	**+27**	2.47	10	**-56**	Yes
Skin	5.10	305	**-37**	4.01	102	**-28**	Yes
Musculoskeletal	8.22	1041	+2	5.24	168	-6	Yes
Congenital Anomolies	6.24	457	**-23**	5.05	101	-10	Yes
Perinatal	9.84	310	**+22**	2.49	11	**-55**	Yes
Ill-defined	2.68	558	**-67**	2.81	224	**-50**	No
Injury and Poisoning	9.09	2975	**+13**	2.42	86	**-57**	Yes

Table 4: Comparison of MCSP scores using the embeddings from Beam et al when considering all relevant embeddings, and only those that are overlapping with the other sets of embeddings. Includes the mean MCSP score for a system, number of examples per system, and the percentage difference vs mean. Significant differences are shown in orange/blue for above/below at $p < 0.05$. The final column is whether the scores of all embeddings vs overlapping embeddings are expected to be from a different population with $p < 0.05$.

lar, both employing discounted cumulative gain to score whether neighbours are related according to differing known relations. Their results are broadly in line with Beam's methodology, which uses similar known relations but instead judges them against a bootstrap of random relations.

Of note, besides Yu's method, all other methods can vary depending on what known relation is used. For instance, Beam et al's method as implemented in our project evaluates known relations between drugs and the diseases they treat or prevent, but it could instead be changed to evaluate whether two concepts are in the same medical system. This is another variable affecting our results. One can understand our results as testing just one configuration of a method, as opposed to evaluating them more broadly.

4.4 Lessons Learned

The main lesson from this project was to establish an evaluation framework ahead of time. The many variables - set of embeddings, medical system, evaluation methods and their known relations - make it difficult to sort out definitive results. This challenge could likely be alleviated by learning more statistics such as ways to evaluate multivariate variability. A smaller project using only one set of embeddings could have helped focus the results, but would remove some potential contribution.

5 Future Work

Improving the current project: There are multiple ways to improve the current project. Different statistical analyses, including non-parametric methods, could be used to consider all raw scores. This would replace the current analysis which often calculates mean scores, losing information such as examples per system. These different statistical methods may allow aggregation of evaluation model results, allowing a more definitive answer as to whether embedding sets or medical systems lead to significant differences.

It was difficult to compare evaluation methods due to not knowing what results to expect. A possible workaround is to construct negative controls, for both systems and embeddings. For instance, random ranges of ICD9 codes could be used to construct such systems, and randomized CUIs used for such embeddings. Indeed, including the results of these controls could make other results more interpretable. Positive controls would likely take more work to construct, but could also be possible.

Currently, the dictionary between UMLS CUIs and ICD9CM codes contains around 40,000 entries. As such, many CUIs could not be used, despite representing concepts very related to those in the dictionary. Generating a larger dictionary automatically would be useful, and could be feasible given the requirement that they only

be labelled into medical systems and their wide range of ICD9 codes. ICD9 is itself an old system, and the newer ICD10 systems could instead be used, or the ICD11 system about to be released this year.

As discussed previously, a possible confounder is whether embeddings from a given medical system perform worse due to containing rare conditions. We could investigate this only comparing embeddings that represent common conditions. We intended this to be part of the current project, but deferred it. Frequencies of ICD9 codes are not readily available except within public health databases, which are considered sensitive and require formal applications to access. This likely could not be acquired in the time-frame of this project, though they represent an interesting avenue for future work.

Possible extensions: Evaluating Zhang et al's domain adaptation-trained embeddings in this project would help quantify this technique's potential benefit, but we were unable to obtain their embeddings. It would be interesting to carry DA out on embeddings from a poorly performing medical system. Or, for a larger project, medical-field-specific embeddings could be trained, and compared to the ones used in this project, in order to determine the scale of possible improvement.

Finally, in this project we quantify embedding quality by their geometric qualities against known relations. A more 'real-world' evaluation could be attempted carrying out actual NLP tasks on documents from different medical systems to understand their relative performance from a more applied perspective. For instance, we could use perform NLP tasks using the embeddings on articles about conditions from the various medical systems, using Wikipedia or a medical encyclopaedia like UpToDate. Or, to be even more applied, NLP tasks could be evaluated on medical documents produced by physicians of different specialities, an opportunity we may soon have access to from an ongoing project.

References

[Beam et al.2018] Andrew L. Beam, Benjamin Kompa, Inbar Fried, Nathan P. Palmer, Xu Shi, Tianxi Cai, and Isaac S. Kohane. 2018. Clinical Concept Embeddings Learned from Massive Sources of Multimodal Medical Data. *arXiv:1804.01486 [cs, stat]*, April.

[Bodenreider2004] Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): Integrating biomedical terminology. *Nucleic Acids Research*, 32(Database issue):D267–D270, January.

[Choi et al.2016] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. 2016. Learning Low-Dimensional Representations of Medical Concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41–50, July.

[De Vine et al.2014] Lance De Vine, Guido Zuccon, Bevan Koopman, Laurianne Sitbon, and Peter Bruza. 2014. Medical Semantic Similarity with a Neural Language Model. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1819–1822, New York, NY, USA. ACM.

[Forbush et al.2013 3 18] Tyler B. Forbush, Adi V. Gundlapalli, Miland N. Palmer, Shuying Shen, Brett R. South, Guy Divita, Marjorie Carter, Andrew Redd, Jorie M. Butler, and Matthew Samore. 2013 -3- 18. "Sitting on Pins and Needles": Characterization of Symptom Descriptions in Clinical Notes". *AMIA Summits on Translational Science Proceedings*, 2013:67–71.

[Lee et al.2018] Hee-Jin Lee, Yaoyun Zhang, Kirk Roberts, and Hua Xu. 2018. Leveraging existing corpora for de-identification of psychiatric notes using domain adaptation. *AMIA Annual Symposium Proceedings*, 2017:1070–1079, April.

[McCoy et al.2016] Thomas H. McCoy, Victor M. Castro, Ashlee M. Roberson, Leslie A. Snapper, and Roy H. Perlis. 2016. Improving Prediction of Suicide and Accidental Death After Discharge From General Hospitals With Natural Language Processing. *JAMA psychiatry*, 73(10):1064–1071, October.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, January.

[Minarro-Giménez et al.2014] José Antonio Minarro-Giménez, Oscar Marín-Alonso, and Matthias Samwald. 2014. Exploring the application of deep learning techniques on medical text corpora. *Studies In Health Technology And Informatics*, 205:584–588.

[Pakhomov2018] Serguei Pakhomov. 2018. Semantic Relatedness and Similarity Reference Standards for Medical Terms, May.

[Slee1978] Vergil N. Slee. 1978. The International Classification of Diseases: Ninth Revision (ICD-9). *Annals of Internal Medicine*, 88(3):424, March.

[Yu and Cai2013] Sheng Yu and Tianxi Cai. 2013. A Short Introduction to NILE. *arXiv:1311.6063 [cs]*, November.

[Yu et al.2017] Zhiguo Yu, Byron C. Wallace, Todd Johnson, and Trevor Cohen. 2017. Retrofitting Concept Vector Representations of Medical Concepts to Improve Estimates of Semantic Similarity and Relatedness. *arXiv:1709.07357 [cs]*, September.

[Zhang et al.2018] Yaoyun Zhang, Hee-Jin Li, Jingqi Wang, Trevor Cohen, Kirk Roberts, and Hua Xu. 2018. Adapting Word Embeddings from Multiple Domains to Symptom Recognition from Psychiatric Notes. *AMIA Summits on Translational Science Proceedings*, 2017:281–289, May.

6 Appendix

This project relies on around 1000 lines of code, which seems infeasible to print out.

As such, over the following pages, I will include code from the three analyses I wrote from scratch. However, this is by no means the extent of the new programming required for the project.

Full code of the project is available on my Github. However, I would rather not make it public. I'm happy to add Giuseppe or Linzi as collaborates so they can view it. In my email of the report to Giuseppe, I'll also attach a zip of the repo with the code.

Listing 1: Code implementing Yu et al's Analysis

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 23 16:16:47 2019

@author: jjnun
"""

def get_new_sysvec_by_system(filenames_type, icd9_systems, cui_to_icd9_dicts, results):
    """ For each medical system, calculates a repersentative vector based on the
    mean of normalized vecotrs repersenting medical conditions in the system.
    Then caluclates the percentage of drugs known to treat or prevent a disease in
    this system who are closest to the correct system's vector. For drugs that
    treat/prevent conditions in n>1 systems, scores as a correct prediction if
    correct system vector is one of closest n system vectors.
    """

    filename_to_embedding_matrix, idx_to_cui, cui_to_idx, cui_to_icd9_types = generate_overlapping_s

    # Obtain dictionary between cuis and the ICD9 disease systems they're part of/related to
    cui_to_systems = get_cui_to_systems(cui_to_icd9_types, icd9_systems)

    # Get list of ICD9 system names in this analysis
    icd9_systems_names = []

    for row in icd9_systems: icd9_systems_names.append(row[0])
    n_of_systems = len(icd9_systems_names)

    filename_index = 0
    for filename, embedding_type, _ in filenames_type:
#        # Matrix to convert cui to positions in the relevant filename
        embedding_matrix = filename_to_embedding_matrix[filename]

        # Make System Vectors
        systems_sysvec = {}
        systems_n = {}
        systems_correct = {}
        for system in icd9_systems_names:
            systems_sysvec[system] = []
            systems_n[system] = 0.0001
            systems_correct[system] = 0

        for cui in cui_to_idx.keys():
            if cui_to_icd9_types[cui]['icd9_type'] == 'diag':
                cui_vec = embedding_matrix[cui_to_idx[cui],:]
                cui_vec = normalize(cui_vec.reshape(1, -1))[0]
                for system in cui_to_systems[cui]:
                    systems_sysvec[system].append(cui_vec)
                    # Below for generating random set for negative control
                    ##rand_system = random.choice(icd9_systems_names)
                    ##systems_sysvec[rand_system].append(cui_vec)

        for system in icd9_systems_names:
            systems_sysvec[system] = np.array(systems_sysvec[system])
```

```
            systems_sysvec[system] = np.mean(systems_sysvec[system], axis=0)

        # Calculate accuracies using System Vectors.
        for cui in cui_to_idx.keys():
            if  cui_to_icd9_types[cui]['icd9_type'] == 'drug':
                cui_vec = embedding_matrix[cui_to_idx[cui],:]
                #cui_vec = cui_vec/np.linalg.norm(cui_vec) #Normalize
                cui_vec = normalize(cui_vec.reshape(1, -1))[0]
                true_systems = cui_to_systems[cui]

                cos_sims = np.zeros(n_of_systems)

               # Generate list of cos similarities with the system vectors
                for i in range(n_of_systems):
                    system = systems_sysvec.keys()[i]
                    system_vec = systems_sysvec[system]
                    cos_sim = cosine_similarity([cui_vec], [system_vec])[0,0]
                    cos_sims[i] = cos_sim

                n = len(true_systems) # Number of systems this cui treats or prevents or 1 if diagn
                pred_systems = [icd9_systems_names[i] for i in np.argsort(cos_sims)[n:]]

                for system in true_systems:
                    systems_n[system] += 1
                    if system in pred_systems: systems_correct[system] += 1
                    #rand_system = random.choice(icd9_systems_names)
                    #if rand_system in pred_systems: systems_correct[system] += 1

        system_index = 0
        for system in icd9_systems_names:
            results[system_index + 1][0] = re.sub(",", "␣", system)
            results[system_index + 1][filename_index + 1] = '%2.2f'
%(100*systems_correct[system]/systems_n[system]) ##, np.std(np.array(systems_dcg[system])))
            results[system_index + 1][-1] = str(int(systems_n[system])) # Number of examples used fo
            system_index += 1
        filename_index += 1

    return results

#-----------------------------------------------------------------------------

def get_yu_umnsrs_cor_by_system(filenames_type, start, end, cui_icd9_tr, cui_icd9_pr, cui_to_icd9):
    """JJN: Calculates the Spearman Correlation Coefficient between UMNSRS ratings a
    nd vector cosines when a pair contains Either a diagnosis in the ICD 9 category,
    or treats or prevents a condition in that ICD 9 category
    """
    filename_to_embedding_matrix, idx_to_cui, cui_to_idx = generate_overlapping_sets_cui(filenames_ty


    print 'Number␣of␣overlapping␣cuis␣between␣embeddings:␣' + str(len(idx_to_cui))

    #umnsrs_filename = 'UMNSRS_relatedness_mod458_word2vec.csv' #Can use judged relatedness instead
    umnsrs_filename = 'UMNSRS_similarity_mod449_word2vec.csv'
    with open(str(data_folder / umnsrs_filename), 'rU') as f:
        umnsrs_rows = f.readlines()[1:]

    # Find all cuis in the overlapping matrix
    cuis = cui_to_idx.keys()

    # Start arrays to store filenames, and to store the comparison values
    filename_all = []
    value_all = []

    # Following code to output which cuis are overlapping between all sets
    ## o = open(str(results_folder / 'overlappingcuis.txt'), 'w')
```

```
## for cui in cuis: o.write(str(cui)+'\n')
## o.close()

# Use a dict to keep track of how many of the UMNSRS comparisons are used in this system
# To avoid extra code, this will be reset with each file, but will all be the same
compares = {}
compares['possible'] = 0 # Total # of UMNSRS that could be used, as have both cuis found in the
compares['total'] = 0 # Total # of UMNSRS comparisons used in this system
compares['diags'] = 0 # Total # with a diagnosis from this system
compares['drugs'] = 0 # Total # with a drug from this system
files_processed = 0 # Only set above for first file, again, same for all

missing_cuis = []

for filename, embedding_type, _ in filenames_type:
    #Contains the unmsrs scores from comparisons that have to do with this system
    unmsrs_scores = []
    #Contains the vector cosine similarity between the embedding vectors
    veccos_scores = []
    # Matrix to convert cui to positions in the relevant filename
    embedding_matrix = filename_to_embedding_matrix[filename]

    for row_str in umnsrs_rows:
        row = row_str.strip().split(',')
        umnsrs_rating = row[0]
        cui_1 = row[4]
        cui_2 = row[5]

        if (cui_1 in cuis) and (cui_2 in cuis):
            if files_processed == 0: compares['possible'] += 1
            in_system_diag_1, in_system_drug_1 = cui_in_system(cui_1, start, end, cui_icd9_tr, c
            in_system_diag_2, in_system_drug_2 = cui_in_system(cui_2, start, end, cui_icd9_tr, c
            if in_system_diag_1 or in_system_drug_1 or in_system_diag_2 or in_system_drug_2:
                # Calculate Cosine similiarity between vectors
                vec_1 = embedding_matrix[cui_to_idx[cui_1],:]
                vec_2 = embedding_matrix[cui_to_idx[cui_2],:]
                cos_sim = cosine_similarity([vec_1], [vec_2])[0,0]

                # Recrod scores
                veccos_scores.append(cos_sim)
                unmsrs_scores.append(float(umnsrs_rating))

                # Keep track of # of UMNSRS comparisons for this system
                if files_processed == 0:
                    in_system_diag = in_system_diag_1 or in_system_diag_2
                    in_system_drug = in_system_drug_1 or in_system_drug_2

                    if in_system_diag:
                        compares['diags'] += 1
                        compares['total'] += 1
                    if in_system_drug:
                        compares['drugs'] += 1
                        compares['total'] += 1
                    if in_system_drug and in_system_diag:
                        compares['total'] -= 1 # Don't double count

        elif cui_1 in cuis:
            missing_cuis.append(cui_2)
        else:
            missing_cuis.append(cui_1)

    # Carryout and append Spearman Rank Correlation
    rho, pval = spearmanr(unmsrs_scores,veccos_scores)
    filename_all.append((filename))
    value_all.append(rho)
```

```
        files_processed += 1

    return filename_all, value_all, compares

#-------------------------------------------------------------------------------

def get_beam_bootstrap_by_systems(filenames_type, icd9_systems, cui_to_icd9_dicts, results):
    """JJN: Using Beam et al's boostrap method, calculates the percentage of known
    drug-disease relations for a given system are within the top 5% of relations
    in a null distribution
    """
    filename_to_embedding_matrix, idx_to_cui, cui_to_idx, cui_to_icd9_types = generate_overlapping_s

    # Obtain dictionary between cuis and the ICD9 disease systems they're part of/related to
    cui_to_systems = get_cui_to_systems(cui_to_icd9_types, icd9_systems)


    # And also build a similiar idx_to_system, and a list of idx that are drugs, or diags
    idx_to_tr_pr_systems = {}
    idx_to_tr_pr_cuis = {}
    drug_idxs = [] # idx's of cuis repersenting a drug
    diag_idxs = [] # '                          ' a diagnosis
    for idx in idx_to_cui:
        cui = idx_to_cui[idx]
        systems = cui_to_systems[cui]
        idx_to_tr_pr_systems[idx] = systems
        if cui_to_icd9_types[cui]['icd9_type'] == 'diag':
            diag_idxs.append(idx)
        elif cui_to_icd9_types[cui]['icd9_type'] == 'drug':
            drug_idxs.append(idx)
            tr_pr_cuis = cui_to_icd9_types[cui]['cuis']
            idx_to_tr_pr_cuis[idx] = tr_pr_cuis
            assert len(tr_pr_cuis) == len(systems), 'While building dictionary, cuis and systems had
        else:
            raise Exception('Each cui used must repersent a ICD9 diagnosis, or drug that treats one')

    # Get list of ICD9 system names in this analysis
    icd9_systems_names = []
    for row in icd9_systems: icd9_systems_names.append(row[0])

    filename_index = 0
    for filename, embedding_type, _ in filenames_type:
#        # Matrix to convert cui to positions in the relevant filename
        embedding_matrix = filename_to_embedding_matrix[filename]

        # Create a null distribution by a bootstrap sample involving a set number
        # of cosine similiarities between random drug and disease pairs
        n_bootstrap = 10000
        null_cos_sims = []
        p_value = 0.05
        for j in range(n_bootstrap):
            rand_drug = random.choice(drug_idxs)
            rand_diag = random.choice(diag_idxs)
            vec_1 = embedding_matrix[rand_drug,:]
            vec_2 = embedding_matrix[rand_diag,:]
            cos_sim = cosine_similarity([vec_1], [vec_2])[0,0]
            null_cos_sims.append(cos_sim)

        # Calculate the threshold for p < 0.05 significance in cosine similarity
        pos_threshold = int(n_bootstrap*p_value)
        sig_threshold = sorted(null_cos_sims)[-1*pos_threshold]
        print 'Done with bootstrap. Have this many examples: ' + str(len(null_cos_sims))
        print 'Significance threshold is: ' + str(sig_threshold)

        systems_n = {}
```

```
        systems_sig = {}

        # Set up
        system_index = 0
        for system in icd9_systems_names:
            systems_n[system] = 0.0000001
            systems_sig[system] = 0
       # Test all drug-relations that have cuis in this system
        for idx in drug_idxs:
            cui_drug = idx_to_cui[idx]
            tr_pr_systems = idx_to_tr_pr_systems[idx]
            tr_pr_cuis    = idx_to_tr_pr_cuis[idx]
            assert len(tr_pr_systems) == len(tr_pr_cuis), 'Length must be same as they correspond'

            for i in range(len(tr_pr_systems)):
                tr_pr_system = tr_pr_systems[i]
                tr_pr_cui = tr_pr_cuis[i]

                # Ignore this treated or prevented disease if it doesn't have a vector repersentatio
                # Not checked until this point as for other analysis we just want to know the systen
                # And our cui_to_icd9 dictionary has more entries than embedings
                if tr_pr_cui in cui_to_idx.keys():
                    systems_n[tr_pr_system] += 1
                    vec_1 = embedding_matrix[cui_to_idx[cui_drug],:]
                    vec_2 = embedding_matrix[cui_to_idx[tr_pr_cui],:]
                    cos_sim = cosine_similarity([vec_1], [vec_2])[0,0]

                    if cos_sim > sig_threshold: systems_sig[tr_pr_system] += 1

        # Store results
        system_index = 0
        for system in icd9_systems_names:
           results[system_index + 1][0] = re.sub(",", " ", system)
           results[system_index + 1][filename_index + 1] = '%2.5f' %(systems_sig[system]/systems_n[s
           results[system_index + 1][-1] = str(systems_n[system]) # Number of examples used for this
           system_index += 1
        filename_index += 1

    return results
```