LIMSI ICD10 coding experiments on CépiDC death certificate statements

Pierre Zweigenbaum and Thomas Lavergne

[1] LIMSI, CNRS, Université Paris-Saclay, F-91405 Orsay, France
pz@limsi.fr,
WWW home page: https://perso.limsi.fr/pz/
[2] LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, F-91405 Orsay, France
lavergne@limsi.fr,
WWW home page: https://perso.limsi.fr/lavergne/

Abstract. We describe LIMSI experiments in ICD10 coding of death certificate statements with the CépiDc dataset of the CLEF eHealth 2016 Track 2. We tested a classifier with humanly-interpretable output, based on IR-style ranking of candidate ICD10 diagnoses. A tf.idf-weighted bag-of-feature vector was built for each training set code by merging all the statements found for this code in the training data. Given a new statement, we ranked candidate codes with Cosine similarity. Features included meta-information and n-grams of normalized tokens. We also prepared an ICD chapter classifier with the same method and used it to rerank the top-k codes (k=2) returned by the code classifier.
For development we focused on mono-code statements and obtained a P@1 of 0.749 increased to 0.778 by chapter reranking. On the test data we returned one code for each statement, leaving multiple code assignment for future work, and obtained a precision, recall and F-measure of (0.7650, 0.5686, 0.6524).

Keywords: ICD10 coding, information retrieval, CépiDc, death certificates, chapter coding, reranking

1 Introduction

Coding in the International Classification of Diseases (ICD) has been the subject of a number of studies in the past (e.g., [9,2]). Until recently, only one shared task had taken it as its target [8], and after ten more years the CLEF eHealth 2016 Task 2 on multilingual information extraction offers a much larger-scale and real-life dataset for ICD10 coding.

The main methods for coding or more generally for concept normalization rely on dictionary-based lexical matching and on machine-learning. In the medical domain, most dictionary-based methods use the UMLS [3] or one of the vocabularies it contains, including the ICD10 classification. For the English language, MetaMap [1] is the best known system and combines the wealth of term variants of the UMLS MetaThesaurus with lexical knowledge including morphological rules from the UMLS Specialist Lexicon to map phrases to UMLS concepts. Language-agnostic methods of approximate term look-up have also been proposed [10].

Koopman et al. [5] studied the classification of Australian death certificates as pertaining to diabetes, influenza, pneumonia and HIV with SVM classifiers based on n-grams and SNOMED CT concepts, and with rules. They also addressed their classification into 3-digit ICD-10 codes such as E10. In another study, the same team [6] trained SVM classifiers to find ICD-10 diagnostic codes for cancer-related death certificates. Because they focused on cancer, they used a first-level classifier to identify the presence of cancer then applied a second-level classifier to identify the specific type of cancer, again as 3-digit ICD-10 codes from C00 to C97. An important difference from the present work is that they targeted the underlying cause of death, i.e., one diagnosis per death certificate, whereas we aim to determine all the diagnoses mentioned in a given death certificate, more precisely at the level of each input statement. Besides, the present work addresses full four-digit ICD-10 codes (e.g., R09.2) instead of three-digit codes (e.g., R09).

The International Classification of Diseases is hierarchical, and previous work has tried to take advantage of this hierarchy to improve classification [7,4]. We have performed limited experiments to leverage the hierarchical structure of the ICD-10 by using an ICD chapter classifier to rerank base results.

The present work addresses the coding of death certificate statements with no restriction on the domain, i.e., into the whole of ICD-10. Rather than aiming at a method that would obtain the best possible results, we were interested in exploring simple, humanly-interpretable vector space representations which would highlight the importance of each word in the classification. We selected an Information Retrieval style method based on the vector space model and Cosine similarity and made experiments with it, reporting a relatively good precision (0.7650) on the official test set. Since we only targeted one code per statement and reserved multi-label classification for further work, we miss the many additional codes present in statements with multiple codes and obtain a limited recall of 0.5686 on the official test set.

We describe how we prepared the source data (Section 2), the methods we tested to find ICD10 codes for input statements (Section 3), ICD10 chapter classification and its use for reranking (Section 4). Experiments and results on the training dataset are presented and discussed along the way. We then provide the results obtained on the test set, perform a short error analysis based on the training set, and discuss perspectives for future work (Section 5).

2 Preparation of the material

CLEF eHealth 2016 participants were provided with the 'Aligned Causes' file (AlignedCauses_2006-2012.csv) which contains 65,843 death certificates split into close to 200,000 diagnostic statements. Each diagnostic statement has associated metadata and zero, one or more ICD-10 codes. In the provided training and test data, statements with multiple codes were repeated on as many lines as they had codes. We considered such repeated statements as one statement, and a large part of our work focused on statements with exactly one code. Besides, we used the resulting list of diagnostic statements independently of each other, without taking into account their grouping into death certificates.

2.1 Basic processing

We processed each diagnostic statement as follows:

- tokenize (French style: NLTK v3.2.1, regular expression);
- remove stop words (French NLTK);
- remove diacritics;
- lower-case;
- stem (Snowball French stemmer in NLTK).

All programs here and in the remainder of this work were written using Python v3.5.

2.2 Hyphenated words

A number of medical words are (often neoclassical) morphological compound words (e.g., *cardiovasculaire*). These words are often written with hyphens (e.g., *cardio-vasculaire*), and sometimes even split (e.g., *cardio vasculaire*). To normalize these words, we adopted the following strategy:

- Our goal is to replace variants with the split form: we hope this will allow to take better account of the components of these words (e.g., the component *vasculaire* in *cardiovasculaire* will share counts with the free-standing *vasculaire* in *arrêt vasculaire*).
- To build a dictionary which normalizes concatenated forms to split forms,
  - We collect tokens with hyphens in statements.
  - We produce concatenated forms from these tokens.
  - We associate the concatenated form to the split form in the dictionary.
  - Once all forms in the training corpus have been processed, dictionary expansions are examined to find tokens which are themselves an entry in the dictionary: these tokens are replaced with their split form.
- During tokenization, hyphenated forms are split on hyphens. Their components are checked for further splitting based on the above dictionary. Non-hyphenated forms which have an entry in the dictionary are replaced with the associated split form.
- In a few cases, components are not meaningful in French (e.g., the English load word *pace-maker*, or the misspellings *inhalat-ion*, *aort-ique*, *ac-fa*). A small set of exceptions (the four above-mentioned cases) was compiled manually based on an examination of the dictionary built with the training corpus. Instead of being split during tokenization, they are replaced with their concatenated form (*pacemaker*, *inhalation* etc.).

The addition of this processing improved precision by about 1pt in all experiments. For the sake of space, we only present below experiments where this processing is included.

2.3 ICD-10 data

We obtained the ICD-10 hierarchy from the UMLS MRREL table and used it to know which code belongs to which ICD-10 chapter. In some experiments we also used the actual labels of the ICD-10 diagnoses. For this purpose, we used the French ICD10, German Modification, version of 2014, downloaded on 12 Feb 2015.

3 IR-style, vector space representation for coding

We tested information-retrieval type methods as described below.

3.1 Principles

Representation and training

- Each statement of the training corpus is tokenized and normalized as described in Section 2.
- One 'document' is created for each code: all the statements associated with a given ICD-10 code C_i are grouped into one text collecting its tokens $T(C_i) = (t_1, t_2, \ldots t_n)$.
  Statements with more than one code are excluded from our training split: the goal is to gather into our training material only statements which are fully associated with one code ('mono-code' statements).
- A bag-of-words, tf.idf model is computed from the token$\times$document matrix (using the tf.idf implementation in gensim v12.4).

Coding as search

- Each tested document is represented as a bag-of-words with tf.idf weights according to this model.
- This representation is compared to the representations of the training 'documents', which represent the ICD10 codes present in the training corpus. Cosine similarity is used.
- The top-N most similar documents (codes) are returned.

Only the top code is used in the prediction. Additionally, the following codes are computed to evaluate success-at-N metrics (this is detailed in Section Experiments below). Besides, examining the quality of the top-N codes can inform us about the potential relevance of reranking methods which could be applied to these top-N codes.

3.2 Experiments

Base principle: test on last 10,000 statements, train on rest (–10k)* For our experiments, we split the CépiDc training corpus into a training split and a test split. The training split consists of the training corpus except its last 10,000 statements (i.e., the first 185,203 statements). The test split consists of the last 10,000 statements, which were all coded in year 2012.

A tf.idf model was created from the training corpus as explained above. Only mono-code statements were used for training.

Statements in the test split were coded as explained above. For each such statement, we measured the following information: whether the correct code was found among the top N codes (Success@N, or S@N, for $N \in [1 \ldots 5]$, as named for instance in TREC); note that S@1 is equal to P@1 (precision computed by examining only the top returned code), and that we therefore often use the term "precision" to comment success results; for some of the tests, the official scoring program was also applied to the system results and computed Precision, Recall and F-measure based on all statements in the test split (columns P, R, F).

We were mostly interested in evaluating results on mono-code statements, since these are those for which our system is relevant. As we shall see below, they represent 77% of the statements in the test split. However, for information, we also evaluated results on all statements with at least one code (98% of the statements in the test split; the remaining 1.7% have no associated code). These two settings imply a different number of statements submitted to the test.

Table 1 provides this information for the various experiments described below. The experiment name (–10k*) codes the characteristics of the experiment, such as 10k (10,000 statements in the test split), 1 (mono-codes) vs. a (all statements with at least one code), etc. Column Sttmts shows the actual number of statements submitted to the test, then the rest of the columns provide evaluation results as described above. The same statements are evaluated by testing whether the code found has the correct ICD10 chapter (rows Chapter, shown for a subset of the evaluations). The statistical significance of differences between experiments was computed using approximate randomization at $p < 0.01$.

We now turn to the description of a series of experiments performed by adding various features to the representation of statements.

Features = tokens (Table 1, rows –10k1t) In this test, the only features for a training or test statement are its normalized tokens. We test our classifier on the subset of 7669 statements with exactly one code among the 10,000 test statements (mono-code statements): in this setting, one precision point (1%) represents 77 statements. As shown on row –10k1t Codes, 71.1% of these statements obtain a correct code at rank 1, and for 84.1% the top code belongs to the correct ICD10 chapter (row Chapter). Success increases significantly between S@1 and S@2 both for codes (12pt from 0.711 to 0.836) and for chapters (10pt from 0.841 to 0.941).

Use of the standardized form of statements (only available for training) (–10k1ts) Each CépiDc training data line includes a standardized form of the part of the

Table 1. Code classification, one target 'document' per code in the training corpus. Train and test on splits of 185,203 and latter 10,000 statements of the training corpus. –10k1tmb: *add bigrams.* –10k1tmbt: *bigrams and trigrams.* –10k1tmbts: *bigrams and trigrams; add standardized tokens to training documents.* –10k1tmbt4: *1–4-grams and meta-information.* –10k1tmbt45: *1–5-grams and meta-information.* –10k1tmbt4i: *1–4grams and meta-information, use ICD labels.*
–10katmBt: *unigrams, pairs, trigrams and meta-information.* –10katmBT: *pairs and triples.* –10katmbt4: *1–4grams and meta-information.*
S@1 results in bold have a statistically significant difference with the previous row; those in italics have a statistically significant difference with two rows above.

			Success at N					Official evaluation		
Experiment	Level	Sttmts.	S@1	S@2	S@3	S@4	S@5	P	R	F
Evaluation on statements with exactly one code										
–10k1ts	7669	Codes	0.706	0.854	0.890	0.907	0.914	0.7289	0.4201	0.5330
–10k1ts	7669	Chapters	0.835	0.945	0.966	0.974	0.977			
–10k1t	7669	Codes	0.711	0.836	0.886	0.898	0.906	0.7149	0.4102	0.5213
–10k1t	7669	Chapters	0.841	0.941	0.962	0.971	0.975			
–10k1tm	7669	Codes	*0.714*	0.862	0.896	0.911	0.920	0.7185	0.4122	0.5239
–10k1tmb	7669	Codes	**0.734**	0.863	0.897	0.913	0.922	0.7344	0.4234	0.5371
–10k1tmb	7669	Chapters	0.861	0.947	0.968	0.977	0.981			
–10k1tmbt	7669	Codes	**0.738**	0.865	0.898	0.913	0.922	0.7383	0.4256	0.5400
–10k1tmbt	7669	Chapters	0.860	0.948	0.968	0.977	0.980			
–10k1tmbt4	7669	Codes	**0.741**	0.866	0.898	0.914	0.923	0.7420	0.4277	0.5426
–10k1tmbt4	7669	Chapters	0.862	0.948	0.968	0.977	0.980			
–10k1tmbt45	7669	Codes	*0.741*	0.866	0.898	0.914	0.923	0.7418	0.4277	0.5425
–10k1tmbt4i	7669	Codes	0.742	0.864	0.895	0.908	0.918	0.7439	0.4283	0.5436
–10k1tmbt4i	7669	Chapters	0.864	0.947	0.966	0.975	0.979			
–10k1tmB	7669	Codes	0.743	0.869	0.901	0.918	0.927	0.7434	0.4286	0.5437
–10k1tmB	7669	Chapters	0.861	0.952	0.969	0.977	0.982			
–10k1tmBt	7669	Codes	0.744	0.872	0.903	0.917	0.926	0.7445	0.4292	0.5445
–10k1tmBt	7669	Chapters	0.860	0.953	0.969	0.976	0.981			
–10k1tmBT	7669	Codes	**0.749**	0.873	0.903	0.918	0.927	0.7499	0.4323	0.5485
–10k1tmBT	7669	Chapters	0.860	0.953	0.969	0.977	0.983			
Evaluation on statements with at least one code										
–10katmBt	9831	Codes	0.751	0.880	0.912	0.928	0.936	0.7518	0.5557	0.6391
–10katmBt	9831	Chapters	0.869	0.957	0.973	0.980	0.984			
–10katmBT	9831	Codes	0.753	0.882	0.912	0.928	0.937	0.7534	0.5568	0.6404
–10katmBT	9831	Chapters	0.868	0.957	0.973	0.980	0.985			
–10katmbt4	9831	Codes	0.753	0.878	0.910	0.925	0.933	0.7537	0.5571	0.6406
–10katmbt4	9831	Chapters	0.870	0.954	0.972	0.980	0.983			

statement which supported the choice of a target code (`StandardText` field). In the CépiDc coding process, the human coder records in this field the text segment which led to the chosen ICD10 code for the current line, keeping only content words and correcting spelling errors if any. If training is performed on the tokens of the standardized form of a statement instead of the tokens of the original form, performance decreases by 2pt (codes, not shown in table). If training is performed on the tokens of both the original and the standardized form of a statement (rows –10k1ts), performance decreases less, by 0.5pt to

S@1=0.706 (codes), and this is confirmed when evaluating on all statements (not shown in table). We therefore do not pursue with this feature.

Meta-information (–10k1tm) Meta-information, provided in the CépiDc training and test data, was added as follows to training and test bag-of-words:

– Gender: two features, =g0 and =g1
– Age: 5 features corresponding to age ranges based on clusters of initial five-year age ranges : =a0, =a5-20, =a25-35, =a40-65, =a70-; these clusters were manually selected based on a study of the distribution of ages per ICD10 chapter in the training set
– LocationOfDeath: 6 features, one for each LocationOfDeath in the source: =l1, =l2, . . . , =l6
– Type of interval: 5 features, one for each LocationOfDeath in the source: =i1, =i2, . . . , =i5

The addition of these four types of meta-information (rows –10k1tm) slightly improves S@1 (+0.3pt for codes at 0.714, not significant, no improvement for chapters), and S@2 by 2.6pt (codes: 0.862).

Bigrams (–10k1tmb), pairs (–10k1tmB), trigrams (–10k1tmbt), triples (–10k1tmbT), tetragrams (–10k1tmbt4) of tokens were added as follows to training and test bag-of-words:

– After tokens were computed and normalized and stop-words removed,
  • sequences of n consecutive tokens were joined into one n-gram (with space separator);
  • the list of normalized tokens was sorted (in alphabetical order); sets of non-necessarily contiguous tokens were joined into one pair (resp. triple) (with space separator); the order of tokens in pairs or triples is always the alphabetical order. Note that pairs include sorted bigrams, and triples include sorted trigrams.

The addition of bigrams on top of meta-information (rows –10k1tmb) improves S@1 by 2.0pt (codes: 0.734, chapters: 0.861). If pairs are used instead of bigrams (rows –10k1tmB), S@1 improves by another 0.9pt (codes: 0.743) but does not change for chapters (+0.1pt at 0.861).

The addition of trigrams on top of bigrams (rows –10k1tmbt) obtains an improvement of 0.4pt on S@1 (codes: 0.738, chapters: 0.860). The addition of trigrams to pairs (rows –10k1tmBt) does not change S@1 much (codes: 0.744, chapters: 0.860).

The addition of pairs and triples (rows –10k1tmBT) instead of bigrams and trigrams further improves S@1 (codes: 0.749).

Adding tetragrams to bigrams and trigrams increases S@1 by 0.3pt (–10k1tmbt4, codes 0.741). Adding 5-grams (–10k1tmbt45) does not obtain a further increase.

ICD10 labels (–10k1tmbt4i) Some ICD10 codes present in the test corpus may occur rarely or not at all in the training corpus. To make sure that every ICD10 code is known to the trained model, we add ICD10 terms as additional training material. They are added to the ICD code 'documents':

– The label of each ICD10 code is tokenized and normalized as described in Section 2.
– One 'document' is created for each code: if multiple labels are available for a code, they are pasted into one such document.
– These 'documents' are merged with those obtained from the training corpus.
– A bag-of-words, tf.idf model is computed from the token×document matrix as explained in Section 3.1.

Added to one of the best configurations (1–4-grams, meta-information), this does not bring a significant change S@1 (+0.1pt = 0.742).

Testing on statements with at least one code (–10ka)* All tests until now were performed on the mono-code statements of our test split. Our system is designed to produce one code per statement, whatever the expected number of codes. In the case of a multi-code statement, the evaluation considers this code correct if it matches any of the expected codes. Testing on all statements with at least one code therefore results in a slightly higher precision: for instance, with 1–4-grams and meta-information (–10kambt4), +1.2pt for codes and +0.8pt for chapters. Differences among the top performing features are not significant anymore in this setting.

Official evaluation program We also applied the official evaluation program to the same system output, comparing it to the full set of 10,000 statements in our test split. Its precision is consistent with Success@1. It adds the measurement of recall, which is necessarily impaired when our system is run only on the 7669 mono-code subset of the test split, and which is harmed anyway because our system returns one code per statement even for multi-code statements. This results in an F-measure of 0.640 for *–10katmbt4* (1–4-grams and meta-information), which we retain for further experiments.

4 Chapter classification and reranking

We have seen that success increases sharply from S@1 to S@2, suggesting that reranking the top proposed codes could lead to improvements. For instance, finding the ICD10 chapter for a statement, if precise enough, could help rerank candidate codes.

4.1 Chapter classification as search

Therefore, we created an ICD10 chapter classifier in the same way as the ICD10 code classifier:

- Each tested document is represented as a bag-of-words with tf.idf weights according to this model.
- This representation is compared to the representations of the training 'documents', which represent the ICD10 *chapters* present in the training corpus. Cosine similarity is used.
- The top-N most similar documents (*chapters*) are returned.

Only the top code is used in the prediction (the following codes are only computed to evaluate success at N metrics).

Experiments Table 2 shows the results obtained for chapter classification on our test split. Using the best features found for codes (1–4-grams and meta-information), chapter classification reaches a S@1 of 0.873 (mono-codes) or 0.882 (statements with at least one code), which looks reasonably good enough to attempt to use it for reranking.

Table 2. Chapter classification: IR experiments, tf.idf, one target 'document' per *chapter* in the training corpus. Splits: train and test on separate splits of 185,203 and latter 10,000 statements of the training corpus.
–10kC1t: *only test on 7669 mono-code subset of the latter 10,000 statements; features = tokens.* –10kC1tmbt: *add meta-information, bigrams and trigrams* –10kC1tmbt4: *add 4grams.* –10kC1tmBT: *pairs and triples.* –10kCatmbt: *test on all statements with at least one code, features = tokens, meta-information, bigrams and trigrams.* –10kCatmbt4: *add 4grams.* –10kCatmBT: *pairs and triples.*
S@1 results in bold have a statistically significant difference with the previous row; those in italics have a statistically significant difference with two rows above.

			Success at N				
Expe.	Level	Sttmts.	S@1	S@2	S@3	S@4	S@5
Evaluation on statements with exactly one code							
–10kC1t	7669	Chapters	0.809	0.936	0.951	0.954	0.957
–10kC1tmbt	7669	Chapters	**0.872**	0.931	0.944	0.948	0.948
–10kC1tmbt4	7669	Chapters	*0.872*	0.931	0.944	0.948	0.948
–10kC1tmBT	7669	Chapters	0.873	0.924	0.937	0.940	0.941
Evaluation on statements with at least one code							
–10kCatmbt	9831	Chapters	0.881	0.939	0.951	0.954	0.954
–10kCatmbt4	9831	Chapters	0.881	0.939	0.951	0.954	0.954
–10kCatmBT	9831	Chapters	0.882	0.932	0.944	0.947	0.947

4.2 Reranking codes based on chapter classification

We use the results of ICD10 chapter classification to rerank ICD10 code classification:

- Only the top-predicted chapter is used, if any.
- The top N predicted codes are examined:

  - The first code which belongs to that chapter is pushed to top position, if any;
  - Otherwise, code ranking is kept unchanged.

– We experimented with $N \in \{1 \ldots 5\}$ in the training set and found out that a conservative value of $N = 2$ obtained the best results.

Experiments Table 3 shows the results obtained for chapter-reranked code classification on our test split. Using the best features found for codes (1–4-grams and meta-information, or unigrams, pairs and triples, and meta-information), chapter-reranked code classification boosts S@1 by 4.5pt (0.786, mono-codes) or 3.2pt (0.785, statements with at least one code): both differences are significant.

Table 3. Chapter-based code classification reranking: IR experiments, tf.idf. Rerank top-2 codes. Splits: train and test on separate splits of 185,203 and latter 10,000 statements of the training corpus.
–10kB1tmbt: *only test on 7669 mono-code subset of the latter 10,000 statements; features = tokens, meta-information, bigrams and trigrams.* –10kB1tmbt4: *add 4grams.*
–10kBatmbt: *only test on 9831 statements with at least one code of the latter 10,000 statements; features = tokens, meta-information, bigrams and trigrams.* –10kBatmbt4: *add 4grams.*
S@1 results in bold have a statistically significant difference with the previous row.

			Success at N					Official evaluation		
Expe.	Level	Sttmts.	S@1	S@2	S@3	S@4	S@5	P	R	F
Evaluation on statements with exactly one code										
–10kB1tmbt	7669	Codes	0.776	0.865	0.898	0.913	0.922	0.7766	0.4477	0.5679
–10kB1tmbt4	7669	Codes	**0.778**	0.866	0.898	0.914	0.923	0.7792	0.4492	0.5698
–10kB1tmBT	7669	Codes	**0.786**	0.873	0.901	0.917	0.927	0.7866	0.4535	0.5753
Evaluation on statements with at least one code								Eval. on all statements		
–10kBatmbt	9831	Codes	0.782	0.877	0.909	0.925	0.933	0.7824	0.5783	0.6650
–10kBatmbt4	9831	Codes	**0.784**	0.878	0.910	0.925	0.933	0.7845	0.5799	0.6668
–10kBatmBT	9831	Codes	0.785	0.882	0.912	0.928	0.937	0.7857	0.5808	0.6679

5 Results and discussion

5.1 Results

The system was trained with parameters *–10kBatmbt4* (codes reranked with chapters, with 1–4grams and meta-information) on the full training set then applied to the test dataset. It was asked to produce one code per input statement, as on the training set. Table 4 shows the results it obtained. For comparison, we recall in the same table the results obtained with the same settings by training on our training split of the training corpus and testing on our test split of the training corpus.

Table 4. Official results on our test split and on the test set.

	precision	recall	F-measure
Test split (development) (–10kBatmbt4)	0.7845	0.5799	0.6668
Test set	0.7650	0.5686	0.6524

5.2 Discussion

Without surprise, recall and F-measure remain lower than precision, as on the development set, mainly because our system does not include provision for multi-label classification.

We also observe that the precision, recall and F-measure of the system did not change much (about –2pt for P, –1pt for R and –1pt for F) from the development set to the test set: this shows that our system did not overfit the training corpus.

We examined the most frequent false positive codes in our test split as follows. Given the set of statements associated with an ICD code in the gold standard codes, we counted the number of incorrect codes in system results (false positives) for these statements. We ranked them in decreasing order of false positives and show the top 20 in Table 5.

Table 5. Gold standard codes with the largest number of system errors in the test split

Gold code	Correct	Incorrect	% Incorrect	Top-1 system	Top-2 system	Top-3 system
I619	3	102	0.97	I691 (40)	P524 (22)	S062 (19)
I640	3	101	0.97	I64 (38)	I694 (32)	I601 (5)
C349	145	86	0.37	C349 (145)	C799 (11)	J984 (8)
A419	224	80	0.26	A419 (224)	B99 (50)	A415 (9)
J189	194	77	0.28	J189 (194)	J851 (24)	T857 (13)
R092	423	53	0.11	R092 (423)	T798 (13)	R09 (8)
R263	3	53	0.95	Z740 (40)	E46 (3)	R263 (3)
I635	48	52	0.52	I635 (48)	I693 (41)	I678 (3)
I10	86	51	0.37	I10 (86)	I272 (13)	I509 (3)
I509	173	42	0.20	I509 (173)	I971 (9)	N19 (5)
R571	2	42	0.95	T794 (39)	R571 (2)	R578 (1)
I48	84	37	0.31	I48 (84)	I10 (6)	F03 (2)
R999	42	34	0.45	R999 (42)	R99 (13)	C349 (4)
E119	22	34	0.61	E119 (22)	E118 (7)	I10 (6)
I259	117	27	0.19	I259 (117)	I509 (5)	N189 (3)
R53	62	27	0.30	R53 (62)	Z515 (7)	R068 (3)
C787	45	27	0.38	C787 (45)	C786 (7)	C799 (4)
J449	42	27	0.39	J449 (42)	J180 (7)	J961 (5)
G200	14	27	0.66	G20 (17)	G200 (14)	Q871 (3)
T179	1	27	0.96	W799 (10)	W78 (5)	W79 (4)

Many of the top false-positive codes are among the most frequent codes (e.g., A419, R092, etc.).

The code with the largest number of errors is I619 (*Hémorragie intracérébrale, sans précision* [Intracerebral haemorrhage, unspecified]). This code frequently occurs (105 times in our test split) and was confused by our system 102 times (97%) with other codes. Among these it was confused 40 times with I691 (*Séquelles d'hémorragie intracérébrale* [Sequelae of intracerebral haemorrhage]). Table 6 shows the vector space representation (the top five features) of I619 and its top-three confused codes. Because of these representations, statements such as *AVC hémorragique* (30 occurrences in the test split), which are represented as 'avc', 'hemorrag', 'avc hemorrag', are more similar to I691 than to I619, although they miss the *sequel* feature which would make them really relevant for I691.

P524 (*Hémorragie intracérébrale (non traumatique) du fœtus et du nouveau-né* [Intracerebral (nontraumatic) haemorrhage of fetus and newborn]) is the second most confused code with I619. The discriminant factor for P524 with respect to I619 is that it is a perinatal diagnosis in Chapter XVI of ICD10. However, age=0 (less than 5 years old), an important feature for that ICD10 chapter, did not make it to these top five features, i.e., it has a lower tf.idf weight than the features displayed in Table 6. Statements such as *hémorragie cérébrale* [Cerebral haemorrhage] or *hémorragie intra-cérébrale* [Intracerebral haemorrhage] and the like, whatever their age metadata, have therefore nearly as strong unigram similarity to I619 as to P524, and have higher bigram similarity to it. The absence in a statement of the feature age=0, which is present in all examples of P524 in the training corpus, does not prevent the IR-style classifier from considering the representation of *hémorragie cérébrale* (with an age different from 0) as more similar to that of P524 (which includes the age=0 feature) than to that of I619.

Finally, S062 (*Lésion traumatique cérébrale diffuse* [Diffuse brain injury]) is the third most confused code with I619. Its top five features also show its strong similarity to I619 and give higher weights to 'cerebral', 'hemorrag', 'intra', 'intra cerebral' than I619. Because of that, statements such as *hématome intra-cérébral* [Intracerebral haemorrhage] or *hématome cérébral* [Cerebral haemorrhage] obtain a representation whose 'cerebral', 'intra', and 'intra cerebral' features get a higher similarity score to S062 than to I619. The discriminant factor for S062 is the fact that in the context of Chapter XIX of ICD10 (traumas, poisoning, etc.), it must be a traumatic lesion. In the CépiDc death certificates, this factor is often present not in the statement itself but in its context, i.e., in the other statements of the same certificate. Contrast for instance *hématome intra-cérébral* in Examples 1 and 2 below, each of which shows an excerpt of a death certificate as annotated in the gold standard data. In Example 1 there is no specific context, therefore it is coded as a cardiovascular diagnosis (I619); in Example 2 the problem in Statement 2 is caused by a fall (*chute* in Statement 3), therefore it is coded as a traumatic injury (S062).

Example 1.
77938;2012;2;75;2;1;détresse respiratoire;3;5;1-1;détresse respiratoire;J960

Table 6. Representation of codes confused with I619 in the test split

Code	ICD10 Label	Representation (top five features with tf.idf)
I619	Hémorragie intracérébrale, sans précision [Intracerebral haemorrhage, unspecified]	('hemorrag', 0.511), ('cerebral', 0.380), ('avc hemorrag', 0.320), ('avc', 0.285), ('intra cerebral', 0.239)
I691	Séquelles d'hémorragie intracérébrale [Sequelae of intracerebral haemorrhage]	('hemorrag', 0.419), ('avc hemorrag', 0.401), ('sequel', 0.388), ('avc', 0.363), ('sequel avc', 0.317)
P524	Hémorragie intracérébrale (non traumatique) du fœtus et du nouveau-né [Intracerebral (nontraumatic) haemorrhage of fetus and newborn]	('cerebral', 0.414), ('hemorrag', 0.411), ('hemorrag cerebral', 0.382), ('hemorrag intra cerebral', 0.289), ('intra cerebral', 0.267)
S062	Lésion traumatique cérébrale diffuse [Diffuse brain injury]	('cerebral', 0.464), ('hemorrag', 0.380), ('intra', 0.305), ('intra cerebral', 0.267), ('hemorrag cerebral', 0.246)

77938;2012;2;75;2;2;Hématome intra-cérébral;NULL;NULL;2-1;hématome intra-cérébral;I619

Example 2.
65397;2012;1;80;2;1;Hypertension intra-crânienne;NULL;NULL;1-1;hypertension intra-crânienne;G932
65397;2012;1;80;2;2;hématome intra-cerebral;NULL;NULL;2-1;hématome intra-cérébral;S062
65397;2012;1;80;2;3;chute de sa hauteur;NULL;NULL;3-1;chute de sa hauteur;W18
[...]

This analysis of the top false positive codes for I619 gives an idea of the types of shortcomings of our method and outlines directions for future work.

5.3 Future work

Improving precision. We observed that some discriminant features were not given sufficient weight in the classification. While using more state-of-the-art IR scores and similarities such as BM25 or language models might improve upon tf.idf and Cosine, we believe that using a more classical machine learning classifier based on individual training examples (instead of global pseudo-documents as we did in an Information Retrieval style) might lead to more important changes. Since our submission, we also experimented with various classifiers, among which Logistic Regression and Support Vector Machines, which obtained a precision of 0.89–0.91 on mono-codes (+15pt) with only tokens as features.

The context of a statement, i.e., the other statements of the same certificate, may provide important information in some cases, as we observed in our error analysis. We plan to incorporate this context as additional features for classification. Another possibility will be to use such features in a reranking stage, as we already started to do with our chapter classifier.

Improving recall. The main cause of the low recall of our classifier is that it only proposes one code for each statement, whereas some statements may lead to multiple codes. A possible direction to address it is to select the top-ranked code, then to greedily remove the tokens which support it, and to iterate on the remaining tokens as long as codes are proposed with sufficiently high confidence.

Adding a more traditional strategy based on simple dictionary matching would be yet another way to identify sequences of diagnoses in a statement.

References

1. Aronson, A.R., Lang, F.M.: An overview of MetaMap: historical perspective and recent advances. J Am Med Inform Assoc 17(3), 229–36 (2010)
2. Blanquet, A., Zweigenbaum, P.: A lexical method for assisted extraction and coding of ICD-10 diagnoses from free text patient discharge summaries. Journal of the American Medical Informatics Association 6(suppl) (1999)
3. Bodenreider, O.: The Unified Medical Language System (UMLS): Integrating biomedical terminology. Nucleic Acids Research 32(Database issue), D267–270 (2004)
4. Kamkar, I., Gupta, S.K., Phung, D., Venkatesh, S.: Stable feature selection for clinical prediction: exploiting ICD tree structure using tree-lasso. J Biomed Inform 53, 277–290 (Feb 2015)
5. Koopman, B., Karimi, S., Nguyen, A., McGuire, R., Muscatello, D., Kemp, M., Truran, D., Zhang, M., Thackway, S.: Automatic classification of diseases from free-text death certificates for real-time surveillance. BMC Med Inform Decis Mak 15, 53 (2015)
6. Koopman, B., Zuccon, G., Nguyen, A., Bergheim, A., Grayson, N.: Automatic ICD-10 classification of cancers from free-text death certificates. Int J Med Inform 84(11), 956–965 (Nov 2015)
7. Perotte, A., Pivovarov, R., Natarajan, K., Weiskopf, N., Wood, F., Elhadad, N.: Diagnosis code assignment: models and evaluation metrics. J Am Med Inform Assoc 21(2), 231–237 (Mar-Apr 2014)
8. Pestian, J.P., Brew, C., Matykiewicz, P., Hovermale, D., Johnson, N., Cohen, K.B., Duch, W.: A shared task involving multi-label classification of clinical free text. In: Biological, translational, and clinical language processing. pp. 97–104. Association for Computational Linguistics, Prague, Czech Republic (June 2007), `http://www.aclweb.org/anthology/W/W07/W07-1013`
9. Wingert, F., Rothwell, D., Côté, R.A.: Automated indexing into SNOMED and ICD. In: Scherrer, J.R., Côté, R.A., Mandil, S.H. (eds.) Computerised Natural Medical Language Processing for Knowledge Engineering, pp. 201–239. North-Holland, Amsterdam (1989)
10. Zhou, X., Zhang, X., Hu, X.: MaxMatcher: Biological concept extraction using approximate dictionary lookup. In: Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence. pp. 1145–1149. PRICAI'06, Springer-Verlag, Berlin, Heidelberg (2006), `http://dl.acm.org/citation.cfm?id=1757898.1758065`