**(1). Get result using ytex.**

 1. Import Min Sook's original blog content(data\_content\_tag\_diabetes\_0821.csv) to Mysql:

The csv file that Min Sook give is not formal. It has multiple lines in a field, and this multiple line use '\n'. But the end of the row is used '\r\n'. It also not always used '"' to enclose a field. When import to Mysql, we can can't just used '\r\n' as the "lines terminated by", because that means using '\r' or '\n' as the terminated char.  First, we replace '\r\n' to a special char(this char is never use in the content, e.g. '\0c' or '`'). then use the char as the lines terminated char.

1. Replace all '\r\n' to '`'

Create table CONTENT\_ORG, then import the csv file to Mysq :

Note: the 'blogId' has to be type 'BIGINT(20)'. The ytex tool need this type.

drop table if exists CONTENT\_ORG;

CREATE TABLE CONTENT\_ORG (

`blogId` BIGINT(20) DEFAULT NULL, /\*----blog id \*/

`post\_hashtag` varchar(300) DEFAULT NULL,

`blog\_name` varchar(200) DEFAULT NULL,

`text\_link\_title` varchar(500) DEFAULT NULL,

`text\_content` varchar(10000) DEFAULT NULL

);

/\*load result in to the table\*/

load data local infile 'C:\\fsu\\ra\\UmlsTagger\\data\\data\_content\_tag\_diabetes\_0821.csv'

into table CONTENT\_ORG

fields terminated by ','

enclosed by '"'

lines terminated by '`'

;

**3. Use the ytex tool to get the annotated result. run cTAKES following the ytex document. The parameters for the DBCollector is**

select distinct blogId INSTANCE\_ID from umls.CONTENT\_ORG

select text\_content note\_text from umls.CONTENT\_ORG where blogId = :instance\_id limit 1

Then run cpe, and if you are lucky you will successful. The result will be write to Mysql.

4. Collect the result we interested in.

create table content\_tag\_ytex as

select a.anno\_text, d.instance\_id, c.\* from v\_document\_cui\_sent c

inner join v\_annotation a on c.anno\_base\_id = a.anno\_base\_id

inner join v\_document d on d.document\_id = c.document\_id;

5. get the result relative to T047 only

use umls;

/\*find relation relevant to T047 \*/

drop table if exists tmp\_rel\_diabete\_t047;

create table tmp\_rel\_diabete\_T047 as

select tr.\*, mrsty.tui,mrsty.sty from tmp\_rel\_diabetes tr

inner join mrsty

on (mrsty.cui = tr.cui1 or mrsty.cui = tr.cui2) and mrsty.tui = 'T047';

use ytex;

drop table if exists content\_tage\_ytex\_T047;

/\* find the cui relevant to the T047. take care to the cui1 and cui2 mapping to rel\_cui\*/

create table content\_tag\_ytex\_T047 as (

select cy.\*, tt.rel, tt.rela,tt.cui2 as rel\_cui from content\_tag\_ytex cy

join umls.tmp\_rel\_diabete\_t047 tt

on cy.code = tt.cui1 and tt.tui = 'T047'

) union (

select cy.\*, tt.rel, tt.rela,tt.cui1 as rel\_cui from content\_tag\_ytex cy

join umls.tmp\_rel\_diabete\_t047 tt

on cy.code = tt.cui2 and tt.tui = 'T047'

) ;

6. add primary key 'id'. then add a 'rel\_str' to indicate the relative cui's string, using the preferred string.

alter table content\_tag\_ytex\_T047 add id Int NOT NULL AUTO\_INCREMENT PRIMARY KEY;

alter table content\_tag\_ytex\_T047 add (`rel\_str` varchar(1000) default null);

update content\_tag\_ytex\_T047 as ct

inner join umls.mrconso as con

on con.cui = ct.code and con.stt='PF'

set ct.rel\_str = con.str;

7. Pick only one result for every 'unique term' (blogId+CUI+termOffset), (5967 results)

create table content\_tag\_ytex\_T047\_unique as

select ct.instance\_id as blogId,

ct.anno\_text as target,

ct.code as CUI,

(select sab from umls.mrconso where cui = ct.code limit 1) as SAB,

ct.cui\_text as umlsStr,

(select tui from umls.mrsty where cui = ct.code limit 1) as TUI,

(select sty from umls.mrsty where cui = ct.code limit 1) as styName,

ct.anno\_base\_id as worldIndex,

ct.sentence\_text as sentence,

ct.rel\_cui,

ct.rel\_str,

ct.id

from content\_tag\_ytex\_T047 as ct

inner join (select distinct id from content\_tag\_ytex\_T047 group by instance\_id, code, anno\_base\_id) as temp

on ct.id=temp.id;

**(2). How to import the original tag of the blog's content. This result is produced by our tool.**

* 1. import it with the follow sql statement:
  2. use umls;

/\*DELET ALL non-english record\*/

-- delete from mrconso where lat <> 'ENG';

drop table IF EXISTS CONTENT\_TAG;

/\*create table for the original terms from blogs.\*/

CREATE TABLE CONTENT\_TAG (

`blogId` varchar(40) DEFAULT NULL, /\*----blog id \*/

`target` varchar(300) DEFAULT NULL, /\* ----the term found in the content. It much be found in UMLS too. If a term is not found in UMLS, it will be ignored. \*/

`umlsFlag` varchar(10) DEFAULT NULL, /\* ----If it is found in UMLS. This column is used by hashTags.\*/

`score` float DEFAULT NULL, /\* ----The similarity metic between the term in the content and the string in UMLS.\*/

`CUI` varchar(45) DEFAULT NULL, /\*----- CUI of UMLS\*/

`SAB` varchar(45) DEFAULT NULL, /\* ----- SAB of UMLS\*/

`AUI` varchar(45) DEFAULT NULL, /\*----- AUI of UMLS \*/

`umlsStr` varchar(1000) DEFAULT NULL, /\* ---STR of UMLS mrconso table\*/

`TUI` varchar(45) DEFAULT NULL, /\*------TUI of UMLS MRSTY table \*/

`styName` varchar(45) DEFAULT NULL, /\*------Semantic name of UMLS MRSTY table\*/

`semName` varchar(100) DEFAULT NULL, /\*---semantic group name in SemGroup website\*/

`tagId` int default 0, /\*----If the term match a hash\_tag of the blog, tagId is the index of the tags. if not match any hash\_tag, it is 0.\*/

`wordIndex` int default 0, /\*----the position of the term in the content\*/

`wordIndexInSentence` int default 0, /\*---- the position of the term in the sentence that it is found in. \*/

`sentenceIndex` int default 0, /\*-- the index of the sentence of the target\*/

`targetNorm` varchar(300) default NULL, /\*--- the normalized string of the term.\*/

`tags` varchar(500) default NULL, /\*---- all the hash\_tags of the blog. \*/

`sentence` varchar(1000) default NULL /\* The sentence that the target is found in\*/

);

/\*load result in to the table\*/

load data local infile 'C:\\fsu\\ra\\UmlsTagger\\data\\data\_content\_tag\_diabetes\_0821\_ret.csv'

into table CONTENT\_TAG

fields terminated by ','

enclosed by '"'

lines terminated by '\n'

ignore 1 lines;

**3. How to find the relationship about 'diabetes'?**

* 1. In umls schema, find all AUI that is relevant to diabetes.
  2. /\*find all AUI that is relevant to diabetes\*/

create table tmp\_diabetes as select distinct AUI from **mrconso** where LAT='ENG' AND (str like '%diabetes%' or str like '%type 1 diabetes%' or str like '%type 2 diabetes%') ;

* 1. Join with table 'mrrel' to file all the relationship about 'diabetes'.

/\*find all relationship that is relevant to diabetes\*/

create table tmp\_rel\_diabetes

as select r.\*

from mrrel r inner join tmp\_diabetes d

on d.aui = r.aui1 or d.aui = r. aui2 ;

**4. How to find subset of the tag result that is relevant to the diabetes?**

Join the tag resutl with the table above 'tmp\_rel\_diabetes ' on the column 'AUI'.

/\*find all tags that has relationship with diabetes\*/

create table content\_tag\_diabetes as select distinct c.\*, r.cui1,r.cui2,r.aui1,r.aui2,r.REL, r.RELA from CONTENT\_TAG c inner join tmp\_rel\_diabetes r

on c.aui = r.aui1 COLLATE utf8\_unicode\_ci or c.aui = r.aui2 COLLATE utf8\_unicode\_ci;

 5. Delete part of the relationship from content\_tag\_diabetes :

/\*delete part of the relationship type\*/

delete from content\_tag\_diabetes where REL = 'SIB' or REL = 'XR' ;

6. Filter the tag result.

1. Add a primary key 'id' for the table content\_tag\_diabetes , we need it to pick the first row if there multiple row for a condition.

/\*add a atuo increment primary key for next step. \*/

alter table content\_tag\_diabetes add id Int NOT NULL AUTO\_INCREMENT PRIMARY KEY;

2. Add a column 'rel\_str' to table, because we want to know the term that is relevant to current target term.

/\*add a column of 'rel\_str' to the result table. it indicates the 'str' of an AUI has relationship with current AUI \*/

alter table content\_tag\_diabetes add rel\_str varchar(1000);

/\*Find out the 'str' of aui that related to current aui. \*/

update content\_tag\_diabetes as cd set cd.rel\_str = (select con.str from mrconso con where con.aui = cd.aui1) where cd.aui1 <> cd.aui COLLATE utf8\_unicode\_ci;

update content\_tag\_diabetes as cd set cd.rel\_str = (select con.str from mrconso con where con.aui = cd.aui2) where cd.aui2 <> cd.aui COLLATE utf8\_unicode\_ci;

3. Pick only one result for a 'unique' term.

Let's define the 'unique term' first: It is a term that found in a position of a unique blog. It means that the  blogid+wordIndex fields of a 'unique term' is unique.

condition: If a 'unique term' match to a CUI, and this term's records for this CUI have multiple rows in the result, we chose the row with highest 'score'.

/\*add a atuo increment primary key for next step. \*/

**alter table content\_tag\_diabetes add id Int NOT NULL AUTO\_INCREMENT PRIMARY KEY;**

/\*pick the first row for the same 'original result' based on blogid+wordIndex+CUI\*/

drop table if exists content\_tag\_diabetes\_unique;

create table content\_tag\_diabetes\_unique as

select cd.\* from content\_tag\_diabetes as cd

inner join (select id from content\_tag\_diabetes group by blogId, CUI, wordIndex order by score desc) as cp

on cp.id = cd.id;

4. add a rel\_cui to the table. it is the cui relative to the current target. It should be one of  the cui1 or cui2.

alter table content\_tag\_diabetes\_unique add rel\_cui varchar(45) default null;

5. find the term that is only relative to 'T047'. (6081 results, run more thant 1 hour.)

create table content\_tag\_diabetes\_T047\_unique as

select cp.\* from content\_tag\_diabetes\_unique as cp

inner join mrsty sty

on cp.rel\_cui = sty.cui COLLATE utf8\_unicode\_ci and sty.tui = 'T047'

;

**(3). Analyzing the different between our result and ytex's result.**

1. find the overlap terms of the two results. (2104 results), use ytex result as the base or use our result as base.

/\* find the overlap content tags both in ytex's result and our result. (blogid, cui, target are same)\*/

drop table if exists content\_tag\_compare\_same;

create table content\_tag\_compare\_same as

select yctu.\* from ytex.content\_tag\_ytex\_T047\_unique as yctu

inner join (

select distinct yctu2.id from ytex.content\_tag\_ytex\_T047\_unique as yctu2

inner join umls.content\_tag\_diabetes\_T047\_unique as uctu

on yctu2.blogId = uctu.blogId and yctu2.cui = uctu.cui COLLATE utf8\_unicode\_ci and yctu2.target = uctu.target COLLATE utf8\_unicode\_ci

) as temp

on yctu.id = temp.id

order by yctu.blogId, yctu.target

;

drop table if exists content\_tag\_compare\_same2;

create table content\_tag\_compare\_same2 as

select uctu.\* from umls.content\_tag\_diabetes\_T047\_unique as uctu

inner join (

select distinct uctu2.id from umls.content\_tag\_diabetes\_T047\_unique as uctu2

inner join ytex.content\_tag\_ytex\_T047\_unique as yctu

on uctu2.blogId = yctu.blogId and uctu2.cui = yctu.cui COLLATE utf8\_unicode\_ci and uctu2.target = yctu.target COLLATE utf8\_unicode\_ci

) as temp

on uctu.id = temp.id

order by uctu.blogId, uctu.target

;

select target, count(\*) as cnt from content\_tag\_compare\_same2 group by target order by cnt desc;

2. find the frequency of the overlap terms in the same result(46 results only)

select target, count(\*) as cnt from content\_tag\_compare\_same group by target order by cnt desc;

select target, count(\*) as cnt from content\_tag\_compare\_same2 group by target order by cnt desc;

3. f**ind the content terms only in ytex's result (3863 result) , and the term frequency**

/\* find the content terms only in ytex's result (3863 result) \*/

drop table if exists content\_tag\_compare\_only\_ytex;

create table content\_tag\_compare\_only\_ytex as

select yctu.\* from ytex.content\_tag\_ytex\_T047\_unique as yctu

where yctu.id not in (

select distinct same.id from ytex.content\_tag\_compare\_same as same

)

order by yctu.blogId, yctu.target

;

select target, count(\*) as cnt from content\_tag\_compare\_only\_ytex group by target order by cnt desc;

/\*203 result\*/

3. find the content terms only in our result,  **and the term frequency**

/\* find the content terms only in our result ( 3938 result) \*/

drop table if exists content\_tag\_compare\_only\_our;

create table content\_tag\_compare\_only\_our as

select uctu.\* from umls.content\_tag\_diabetes\_T047\_unique as uctu

where uctu.id not in (

select distinct same.id from ytex.content\_tag\_compare\_same2 as same

)

order by uctu.blogId, uctu.target

;

select target, count(\*) as cnt from content\_tag\_compare\_only\_our group by target order by cnt desc;

/\*42 result\*/