Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Distàncies *word2vec*

Projecte Targetes Gràfiques (TGA)

Antoni Casas Muñoz
Pol Martín Garcia

Maig 2020

Introducció

Implementació

Hi ha dues versions correctament implementades del projecte, amb resultats finals equivalents.

Aquestes versions també son equivalents al codi seqüencial, trobat al fitxer *main.cpp* en la funció `sequentialSearch()`. Aquesta, donat un vector d'*embeddings*, l'index d'un d'aquests, i un nombre N, troba d'entre tots els *embeddings* del vector els N més semblants a l'*embedding* identificat per l'index.

Algoritme

Qualsevol dels algoritmes implementats per a solucionar aquest problema es basen en 3 parts.

1. Trobar la paraula en el vector d'*embeddings*. Aquest pas sempre és du a terme amb una cerca binària en CPU, per tant no el discutirem en aquest document.
2. Dur a terme el comput de les distancies de cosinus (o similituds de cosinus) entre tots els *embeddings* i l'*embedding* de la paraula cercada.
3. Filtrar els resultats, i escollir les N paraules més semblants (amb major similitud) a la paraula cercada amb les dades calculades, de manera ordenada.

Els punts 2 i 3, es troben tant implementats per a CPU, a *main.cpp*, com per GPU, a *kernel.cu*.

Càlcul de similituds

El càlcul de les distancies o similituds es du a terme a GPU pel kernel `DotProduct()`, el qual calcula el producte escalar amb cadascun dels *embeddings*, i posteriorment en divideix el resultat pel producte de normes.

Això és du a terme movent el *embeddings* de la paraula cercada a memòria *shared*, i posteriorment la usen tots els *threads* del bloc per a dur a terme el producte escalar amb un altre *embedding*.

D'aquesta manera cada *thread* computa una sola similitud.

Filtrat i ordenació

El filtrat per GPU requereix de conèixer les N paraules amb una major similitud. Per a això s'ha dividit el comput d'aquest procés en dues funcions.

La primera, `FirstMerge()`, divideix el vector de similituds resultants en trossos de N elements, els quals son ordenats usant ordenació per inserció donat a que N sempre serà un nombre petit. La ordenació es du a terme *on-place*, de manera que es reutilitza la mateixa memòria per emmagatzemar el resultat.

D'aquesta forma, obtenim el vector de similituds en trossos de N elements internament ordenats. Evidentment, a part d'aquest vector de similituds s'emmagatzema un vector de index a les paraules originals, per no perdre la relació entre valor de similitud i la respectiva paraula.

Seguida i finalment la funció `BotchedMergeSort()` aprofita es segments ordenats per a dur a terme l'ordenació en una reducció del problema. Cada *thread* compara dos dels pedaços de N elements pre-ordenats en un de sol.

Aquesta funció redueix el nombre de similituds a comparar a la meitat per cada crida, i és va usant fins que només resta un sol vector de N elements, el qual identifica les N paraules amb major similitud.

Resultats

Possibles millores