UNIVERSIDAD DE SONORA

UNIDAD REGIONAL CENTRO

LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

INFORME DE SEGUIMIENTO PARA LAS RECOMENDACIONES DEL ÁREA DE INFRAESTRUCTURA Y EQUIPAMIENTO

RESPONSABLE: Donald José Rodríguez Úbeda

PERÍODO: 30/11/19 - 31/12/19

Licenciatura en Ciencias de la Computación

Área

9. Infraestructura y Equipamiento

Descripción

Sección para registrar las evidencias correspondientes al área de Infraestructura y Equipamiento proporcionada por CONAIC para la evaluación de la Licenciatura en Ciencias de la Computación en la Universidad de Sonora.

Recomendación

9.1.1 Mejorar los centros de estudiantes

PLAZO: 01/07/19 - 01/12/19

Metas y acciones

9.1.1.1. Mejorar el centro de cómputo de la licenciatura

9.1.1.1.1. Arreglar las computadoras que no sirven

9.1.1.1.2. Comprar nuevas computadoras

Evidencia

TÍTULO: Se compraron nuevas computadoras

ACCIÓN:
9.1.1.1.2. Comprar nuevas computadoras

Descripción
Se compraron 10 computadoras de la marca HP con pantallas de 24 pulgadas

Fecha: 01/12/19

ANÁLISIS LÓGICO
III SERIE: RESOLUCIÓN PROPOSICIONAL
pedro.loera@unison.mx

Análisis de validez, Formas normales (FNC y FND), forma clausal y resolución proposicional. II serie de ejercicios.

1- **Obtener la FND de:**
   a) $\big(p \to (p \to q)\big) \vee p \vee q$

   b) $\neg(p \vee \neg q) \wedge (\neg r \vee s)$

   c) $\neg(p \to q) \vee p \vee q$

   d) $\big((p \to q) \to r\big) \to s$

   e) $\neg(p \to q) \leftrightarrow p \vee r$

   f) $((\neg p \wedge q) \vee (p \vee \neg q)) \to (\big(p \to (q \vee r)\big) \to (p \to r))$

   g) $\big(p \wedge (q \to p)\big) \to p$

   h) $(\neg p \to q) \to \big((p \wedge \neg q) \vee r\big)$

   i) $\big(p \to (p \to q)\big) \vee p \vee q$

   j) $\neg(p \vee \neg q) \wedge (\neg r \vee s)$

   k) $\neg(p \to q) \vee p \vee q$

   l) $\big((p \to q) \to r\big) \to s$

2- **Obtener la FNC de:**
   a) $\neg(p \to q) \vee p \vee q$

   b) $\big((p \to q) \to r\big) \to s$

   c) $\neg(p \to q) \leftrightarrow p \vee r$

d) $((\neg p \wedge q) \vee (p \vee \neg q)) \rightarrow ((p \rightarrow (q \vee r)) \rightarrow (p \rightarrow r)$

e) $\big(p \wedge (q \rightarrow p)\big) \rightarrow p$

f) $(\neg p \rightarrow q) \rightarrow \big((p \wedge \neg q) \vee r\big)$

g) $\big(p \rightarrow (p \rightarrow q)\big) \vee p \vee q$

h) $\neg(p \vee \neg q) \wedge (\neg r \vee s)$

Determinar la validez de la siguientes formulas mediante los métodos de análisis de validez de razonamientos vistos y escribir la forma clausal de cada una de ellas.

a) $\big((\neg p \wedge q) \vee (p \vee \neg q)\big) \rightarrow \Big(\big(p \rightarrow (q \vee r)\big) \rightarrow (p \rightarrow r)\Big)$
b) $\big(p \wedge (q \rightarrow p)\big) \rightarrow p$
c) $(\neg p \rightarrow q) \rightarrow ((p \wedge \neg q) \vee r)$
d) $(A \wedge B) \vee R \rightarrow \neg A \vee \neg R$
e) $((P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \vee Q) \vee (\neg Q \wedge \neg R)$
f) $(A \wedge B) \rightarrow C,\ A \wedge \neg C \Rightarrow \neg B$
g) $(A \wedge) \vee R \rightarrow \neg A \wedge \neg R$
h) $(A \wedge B) \rightarrow D \wedge F,\ \neg D \Rightarrow \neg F \rightarrow \neg B$
i) $(A \vee B \vee C) \wedge (\neg A \vee \neg B) \wedge (A \vee C) \wedge (A \vee \neg B) \wedge (A \vee C \vee \neg B) \wedge (\neg A \vee \neg C)$=F
j) $(A \vee B \vee C) \wedge \neg A \wedge (B \vee \neg C) \wedge \neg C(\neg B \vee A) \wedge (\neg B \vee C \vee D) \wedge (D \vee \neg C) = F$

Formalizar el siguiente razonamiento, pasar a la forma clausal y hacer el análisis de validez mediante contradicción y resolución proposicional:

"Todo número entero o es primo o es compuesto. Si es compuesto, es un producto de factores primos, y si es un producto de factores primos, entonces es divisible por ellos. Pero si un numero entero es primo, no es compuesto, aunque es divisible por sí mismo y por la unidad, y consiguientemente, también divisible por números primos. Por tanto, todo número entero es divisible por números primos.

Ayuda:

p: Todo número entero es primo.

q: Todo número entero es compuesto.

r: Es un producto de factores primos.

s: Es divisible entre factores primos.

t: Es divisible por sí mismo y por 1.

$F: \{p \vee q, q \rightarrow r, r \rightarrow s, p \rightarrow (\neg q \wedge t \wedge s)\} \Rightarrow s$

Formalizar el siguiente razonamiento, pasar a la forma clausal y hacer el análisis de validez mediante contradicción y resolución proposicional:

"El secretario de Economía y Hacienda ha hecho las siguientes declaraciones:"

A la prensa: "Si los impuestos suben, la inflación bajara, si y solo sí; no se devalúa el peso."

A la radio: "si la inflación baja o si el peso no se devalúa, los impuestos no subirán."

A la televisión: "O bien baja la inflación y se devalúa, el peso, o bien los impuestos deben subir."

Como consecuencia, publica un informe en el que asegura: "los impuestos deben subir, pero la inflación bajara y el peso no se devaluara."

¿fue consecuente con sus declaraciones a los medios de comunicación?

Considere las proposiciones simples:

p: Los impuestos suben.

q: La inflación baja.

r: Se devalúa el peso.

$F1 = p \rightarrow (q \leftrightarrow \neg r)$

$F2 = (\neg q \wedge r) \vee p$

$F3 = (p \vee q) \wedge r)$

$Q = p \wedge q \wedge \neg r$

$R: \{F1, F2, F3\} \Rightarrow Q$

Formalizar el siguiente razonamiento, pasar a la forma clausal y hacer el análisis de validez mediante contradicción y resolución proposicional:

"No llora, ríe. Si no llora, ríe solo si trae un juguete. Nunca trae un juguete cuando se está riendo si no come un caramelo. Luego come un caramelo."

Teniendo las proposiciones simples:

p: Llora

q: Ríe

r: Trae un juguete

s: Come un caramelo

tenemos el razonamiento:

$R: \{\neg p \wedge q, \neg p \rightarrow (q \rightarrow r), \neg s \rightarrow (q \rightarrow \neg r)\} \Rightarrow s$

Pasando a forma clausal

$c = \{\neg p, q, p \vee \neg q \vee r, \neg s \vee r, \neg s \vee r, s\}$

Formalizar el siguiente razonamiento, pasar a la forma clausal y hacer el análisis de validez mediante contradicción y resolución proposicional:

"Cuando salgo sin paraguas llueve. Cuando está despejado no llueve. Según el hombre del clima maña estará despejado o habrá niebla. De cualquier manera, saldré sin paraguas. Entonces mañana, además de llover habrá niebla."

Tenemos las proposiciones simples:

p: Salgo sin paraguas

q: Llueve

r: Esta despejado

s: Habrá niebla

El razonamiento es el siguiente:

$$R: \{p \rightarrow q, r \rightarrow \neg q, r \vee s, p\} \Rightarrow q \wedge s$$

En forma clausal.

$$C = \{\neg p \vee q, \neg r \vee \neg q, r \vee s, p, q, s\}$$

Todos los ejercicios fueron tomados de la red.

PEDRO I LOERA BURNES
DEPTO. DE MATEMÁTICAS
UNIVERSIDAD DE SONORA

Evidencia

Título: Se arreglaron mil computadoras

Acción:
9.1.1.1.1. Arreglar las computadoras que no sirven

Descripción:
Se arreglaron muchísimas pero muchas computadoras en el centro de cómputo de la licenciatura en ciencias de la computación.

Adicionalmente, se le dio servicio a todos los ordenadores de escritorio de los maestros del edificio 3K-4.

Fecha: 20/12/19

UNIVERSIDAD DE SONORA
DIVISIÓN CIENCIAS EXACTAS Y NATURALES

Tema:
Sistema de recomendación con Apache Spark

Materia:
Tópicos avanzados de ciencias computacionales

Profesor:
Juan Pablo Soto Barrera

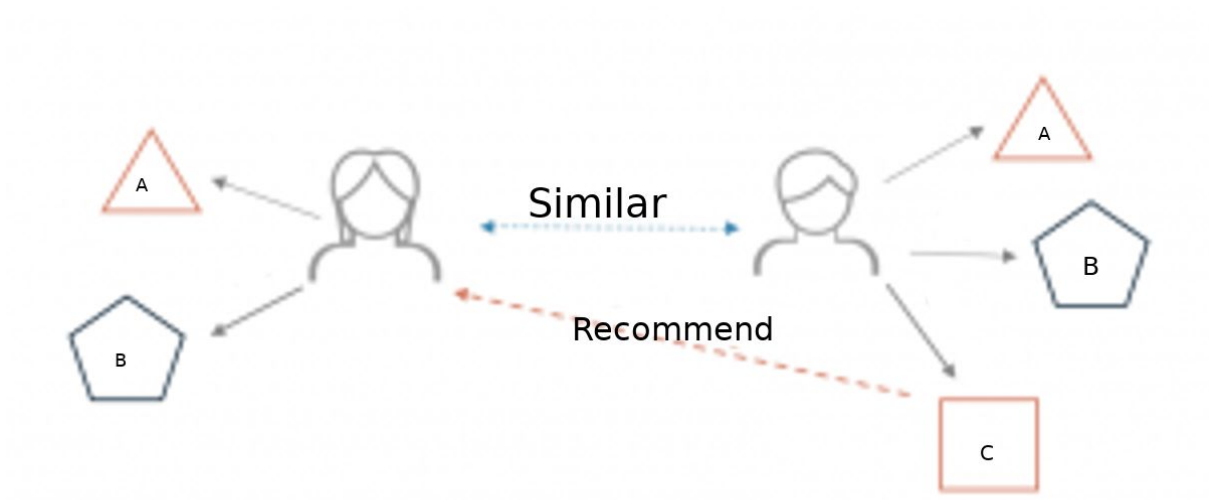
Alumno:
Raúl Francisco Pérez Rodríguez

Introducción

El contenido de este documento es la explicacion y implementación de un sistema de recomendación de tipo filtro colaborativo, donde se recomendaran películas a un nuevo usuario dependiendo del conjunto de películas que haya valorado previamente. Primero se verá una breve explicación de que es un sistema de recomendación y los tipos de sistemas de recomendación, por último se realizará la implementación del mismo utilizando Apache Spark con el API de python.

Sistema de recomendación

Un sistema de recomendación es un algoritmo cuyo objetivo es proporcionar la información más relevante a un usuario mediante el descubrimiento de patrones en un conjunto de datos. El algoritmo clasifica los elementos y muestra al usuario los elementos que calificaria con buena valoración.



Por ejemplo, dos usuarios califican con buena valoración las películas A y B. Cuando esto sucede, se calcula el índice de similitud de estos dos usuario y dependiendo de la puntuación, el sistema puede recomendar la película C al otro usuario, ya que el sistema detectó que esos dos usuarios son similares.

Diferentes tipos de sistemas de recomendación

Los tipos de sistemas de recomendación más comunes son los sistemas de recomendación **basados en contenido** (content based) y de **filtro colaborativo** (collaborative filtering).

En el de filtro colaborativo, el comportamiento de un grupo de usuarios se utiliza para hacer recomendaciones a otros usuarios. La recomendación se basa en la preferencia de otros usuarios. Un ejemplo sería recomendar una pelicula a un usuario basándose en el hecho de que un amigo le gusto la pelicula.

Hay dos tipos de modelos colaborativos. Métodos basados en memoria y métodos basados en modelo. La ventaja de las técnicas basadas en memorias es que son fáciles de implementar y las recomendaciones resultantes son a menudo más fáciles de explicar. Se dividen en dos:

Filtro colaborativo basado en el usuario (User-based collaborative filtering): En este modelo, los productos se recomiendan a un usuario en función de que productos le gustan y han gustado a usuarios similares. Por ejemplo, si a A y B le gustan las mismas películas y sale una nueva película que le gusta a A, entonces se puede recomendar la película a B.

Filtro colaborativo basado en elementos (Item-based collaborative filtering): Estos sistemas identifican elementos similares según las calificaciones de los usuarios. Por ejemplo, si los usuarios A, B y C otorgaron una calificación de 5 estrellas a los libros X e Y, luego cuando un usuario D compra el libro X, también obtienen una recomendación para comprar el libro Y, porque el sistema identifica que los libros X e Y son similares según las calificaciones de usuarios A, B y C.

Los **métodos basados en modelos** se basan en la factorización matricial y son mejores para tratar la escasez. Se desarrollan utilizando minería de datos, algoritmos de aprendizaje automático para predecir la calificación de los usuarios de elementos sin calificación.

Los sistemas **basados en contenido** utilizan metadatos como género, productor, actor, músico para recomendar artículos como películas o música. Por ejemplo, recomendar Infinity War, porque actúa Vin Disiel y a alguien le gusto Rápido y furioso. Del mismo modo, puedes obtener recomendaciones de música de ciertos artistas porque te gustó su música. Los sistemas basados en contenido se basan en la idea de que si le gustó un determinado artículo es probable que le guste algo que sea similar a él.

Implementación

Se utilizará el conjunto de datos de MovieLens para implementar un sistema de recomendación de filtro colaborativo. El conjunto de datos de MovieLens tiene dos opciones, la pequeña (100 mil ratings) o la completa (21 millones de ratings). Se utilizará la completa para probar el rendimiento de Spark.

Primero que todo hay que importar pyspark y inicializar el contexto para poder usar sus funciones.

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
```

Después hay que cargar el archivo de ratings, el archivo de ratings contiene los siguientes campos: userId, movieId, rating y timestamp. Donde se eliminará el timestamp ya que no se necesitará para el recomendador.

```
complete_ratings_raw_data = sc.textFile('data/full/ratings.csv')
complete_ratings_raw_data_header = complete_ratings_raw_data.take(1)[0]

# Parse
complete_ratings_data = complete_ratings_raw_data \
    .filter(lambda line: line != complete_ratings_raw_data_header)\
    .map(lambda line: line.split(",")) \
    .map(lambda tokens:(int(tokens[0]),int(tokens[1]),float(tokens[2]))).cache()

print("Hay %s recomendaciones en el conjunto de datos completo" %
    (complete_ratings_data.count()))
--------------------------------------------------------------------------------
Hay 27753444 recomendaciones en el conjunto de datos completo
```

Hacemos lo mismo para el archivo de movies, el cual tiene los siguientes campos movieId, title y genres, donde se eliminará el campo de genres, ya que no se utilizara para el recomendador.

```
complete_movies_raw_data = sc.textFile('data/full/movies.csv')
complete_movies_raw_data_header = complete_movies_raw_data.take(1)[0]

# Parse
complete_movies_data = complete_movies_raw_data \
    .filter(lambda line: line!=complete_movies_raw_data_header)\
```

```
    .map(lambda line: line.split(",")) \
    .map(lambda tokens: (int(tokens[0]),tokens[1],tokens[2])).cache()

complete_movies_titles = complete_movies_data.map(lambda x: (int(x[0]),x[1]))

print("Hay %s películas en el conjunto de datos completo" %
    (complete_movies_titles.count()))
```

```
Hay 58098 películas en el conjunto de datos completo
```

Lo siguiente será entrenar el conjunto de datos utilizando ALS (Alternating Least Squares) que se encuentra en la biblioteca MLLIB de Spark.

El conjunto de datos se separaran en entrenamiento y en prueba, 70% y 30% respectivamente. Utilizando la función train, se entrenará el modelo para el conjunto de datos. Una vez el modelo termine de entrenar, se probará la precisión del modelo usando los datos de prueba para ello se usará RMSE para ver el error cometido por el modelo.

```
from pyspark.mllib.recommendation import ALS
import math

iterations = 10
regularization_parameter = 0.1
best_rank = 4

training_RDD, test_RDD = complete_ratings_data.randomSplit([7, 3])

complete_model = ALS.train(training_RDD, best_rank, iterations=iterations,
    lambda_=regularization_parameter)

test_for_predict_RDD = test_RDD.map(lambda x: (x[0], x[1]))

predictions = complete_model.predictAll(test_for_predict_RDD) \
    .map(lambda r: ((r[0], r[1]), r[2]))
rates_and_preds = test_RDD.map(lambda r: ((int(r[0]), int(r[1])),
    float(r[2]))).join(predictions)
error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())

print( 'El error para los datos de prueba con RMSE es %s' % (error))
```

```
El error para los datos de prueba con RMSE es 0.8302071628940553
```

Otra cosa que hay que tener en cuenta es dar recomendaciones de películas con un número mínimo de calificaciones. Para eso, necesitamos contar el número de clasificaciones por película realizadas por los usuarios.

```
def get_counts_and_averages(ID_and_ratings_tuple):
    nratings = len(ID_and_ratings_tuple[1])
    return ID_and_ratings_tuple[0], (nratings, float(sum(x for x in
        ID_and_ratings_tuple[1]))/nratings)

movie_ID_with_ratings_RDD = (complete_ratings_data.map(lambda x: (x[1],
    x[2])).groupByKey())

movie_ID_with_avg_ratings_RDD = movie_ID_with_ratings_RDD \
    .map(get_counts_and_averages)

movie_rating_counts_RDD = movie_ID_with_avg_ratings_RDD.map(lambda x: (x[0],
    x[1][0]))
```

Lo siguiente será hacer una recomendación de películas a un nuevo usuario, por lo tanto, se creará un conjunto de películas valoradas.

```
new_user_ID = 0

# The format of each line is (userID, movieID, rating)
new_user_ratings = [
     (0,260,4), # Star Wars: Episode IV - A New Hope (1977)
     (0,1196,4), # Star Wars: Episode V - The Empire Strikes Back (1980)
     (0,1210,4), # Star Wars: Episode VI - Return of the Jedi (1983)
     (0,88140,3), # Captain America: The First Avenger (2011)
     (0,110102,3), # Captain America: The Winter Soldier (2014)
     (0,122920,4), # Captain America: Civil War (2016)
     (0,122892,3), # Avengers: Age of Ultron (2015)
     (0,122910,4), # Avengers: Infinity War - Part I (2018)
     (0,89745,4), # Avengers, The (2012)
     (0,122916,3) # Thor: Ragnarok (2017)
    ]
new_user_ratings_RDD = sc.parallelize(new_user_ratings)
print( 'Ratings del nuevo usuario: %s' % new_user_ratings_RDD.take(10))
-------------------------------------------------------------------------------
Ratings del nuevo usuario: [(0, 260, 4), (0, 1196, 4), (0, 1210, 4), (0, 88140,
3), (0, 110102, 3), (0, 122920, 4), (0, 122892, 3), (0, 122910, 4), (0, 89745,
4), (0, 122916, 3)]
```

Una vez que tengamos las puntuaciones del nuevo usuario, las uniremos con el conjunto de datos completo para reentrenar el modelo con las nuevas valoraciones.

```
complete_data_with_new_ratings_RDD = complete_ratings_data \
    .union(new_user_ratings_RDD)

new_ratings_model = ALS.train(complete_data_with_new_ratings_RDD, best_rank,
   iterations=iterations, lambda_=regularization_parameter)
```

Cuando se termine de entrenar, ya se pueden predecir los valores de las películas que el nuevo usuario no tiene valoradas.

```
new_user_ratings_ids = map(lambda x: x[1], new_user_ratings)

new_user_unrated_movies_RDD = (complete_movies_data
   .filter(lambda x: x[0] not in new_user_ratings_ids).map(lambda
   x:(new_user_ID, x[0])))

new_user_recommendations_RDD = new_ratings_model
    .predictAll(new_user_unrated_movies_RDD)
```

Una vez que se tienen las valoraciones, se buscan los títulos de las películas verificando que cumpla el mínimo de los votos permitido por película, y se ordenan las películas por la valoración. Por último se da un top de las películas que pueden gustarle al nuevo usuario.

```
new_user_recommendations_rating_RDD = new_user_recommendations_RDD.map(lambda x:
(x.product, x.rating))
new_user_recommendations_rating_title_and_count_RDD=
new_user_recommendations_rating_RDD
   .join(complete_movies_titles)
   .join(movie_rating_counts_RDD)

new_user_recommendations_rating_title_and_count_RDD = \
    new_user_recommendations_rating_title_and_count_RDD.map(lambda r:
        (r[1][0][1], r[1][0][0], r[1][1]))

top_movies = new_user_recommendations_rating_title_and_count_RDD
    .filter(lambda r: r[2]>=25).takeOrdered(10, key=lambda x: -x[1])

print('TOP 10 de películas recomendadas:\n%s' %
    '\n'.join(map(str, top_movies)))
--------------------------------------------------------------------------------
TOP 10 de películas recomendadas:
```

```
('Louis C.K.: One Night Stand (2005)', 3.993158993606868, 38)
('DMB (2000)', 3.9907390325178205, 29)
('Hitman Hart: Wrestling with Shadows (1998)', 3.9807323152545466, 33)
('The Second Renaissance Part II (2003)', 3.9279417381965596, 61)
('Down House (2001)', 3.921930985056261, 26)
("Monty Python's Fliegender Zirkus (1971)", 3.9167775760983945, 28)
('Jimmy Carr: Comedian (2007)', 3.8873387247304603, 30)
('Saturday Night Live: The Best of Will Ferrell (2002)', 3.8810049334224272, 26)
('Rick and Morty: State of Georgia Vs. Denver Fenton Allen (2016)',
3.874377809692846, 32)
("Gurren Lagann: Childhood's End (Gekijô ban Tengen toppa guren ragan: Guren
hen) (2008)", 3.8451507719254225, 35)
```

Conclusión

El uso de un sistema de recomendación necesita mucho poder de cómputo y Apache Spark es una buena opción para eso, ya que el modelo de un sistema de recomendación de filtro colaborativo necesita estarse actualizando con las nuevas valoraciones de los usuarios y Apache Spark al poder hacer los cálculos en paralelo con distintos clusters es una gran opción para las grandes empresas que necesitan realizar los cálculos en el menor tiempo posible. Utilizar la librería de MLLIB de Apache Spark hizo el proceso rápido, fácil y entendible, al tener una buena cantidad de funciones que facilitan a la hora de programar.

Referencias

Mwiti, D. (2018). *How to build a Simple Recommender System in Python*. [online] Towards Data Science. Available at: https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d [Accessed 4 Dec. 2018].

Dianes, J. (2018). *Building a Movie Recommendation Service with Apache Spark | Codementor*. [online] Codementor.io. Available at: https://www.codementor.io/jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw [Accessed 4 Dec. 2018].

Spark.apache.org. (2018). *Collaborative Filtering - RDD-based API - Spark 2.1.0 Documentation*. [online] Available at: https://spark.apache.org/docs/2.1.0/mllib-collaborative-filtering.html [Accessed 4 Dec. 2018].

Let It Go

The snow glows white
on the mountain tonight,
not a footprint to be seen.
A kingdom of isolation and it looks
like I'm the queen.
The wind is howling like this
swirling storm inside.
Couldn't keep it in, Heaven knows I tried.
Don't let them in, don't let them see.
Be the good girl you always have to be.
Conceal don't feel, don't let them know.
Well, now they know!

Let it go, let it go!
Can't hold it back anymore.
Let it go, let it go!
Turn away and slam the door.
I don't care what they're going to say.
Let the storm rage on.
The cold never bothered me anyway.

It's funny how some distance,
makes everything seem small.
And the fears that once controlled me,
can't get to me at all
It's time to see what I can do,
to test the limits and break through.

No right, no wrong, no rules for me.
I'm free!

Let it go, let it go.
I am one with the wind and sky.
Let it go, let it go.
You'll never see me cry.
Here I stand, and here I'll stay.
Let the storm rage on.

My power flurries through the air
into the ground.
My soul is spiraling in
frozen fractals all around
And one thought crystallizes like an icy blast
I'm never going back; the past is in the past!

Let it go, let it go.
And I'll rise like the break of dawn.
Let it go, let it go
That perfect girl is gone
Here I stand, in the light of day.

Let the storm rage on!
The cold never bothered me anyway.

Disney FROZEN

SONG LYRICS

Sing-along with the songs from Frozen!

In Summer

Kristoff: Really? I'm guessing you don't have much experience with heat.
Olaf: Nope! But sometimes I like to close my eyes, and imagine what it'll be like when summer does come.

Bees'll buzz, kids'll blow dandelion fuzz
And I'll be doing whatever snow does
in summer.
A drink in my hand,
my snow up against the burning sand
Prob'ly getting gorgeously tanned
in summer.

I'll finally see a summer breeze,
blow away a winter storm.
And find out what happens to solid water
when it gets warm!
And I can't wait to see,
what my buddies all think of
me.
Just imagine how much
cooler I'll be
in summer.

Dah dah, da doo,
uh bah bah bah bah bah boo

The hot and the cold are both so intense,
Put 'em together it just makes sense!

Rrr Raht da daht dah dah dah
dah dah dah dah dah doo

Winter's a good time
to stay in and cuddle,
But put me in summer and I'll be a —
happy snowman!

When life gets rough,
I like to hold on to my dream,
Of relaxing in the summer sun,
just lettin' off steam.

Oh the sky would be blue,
and you guys will be there too
When I finally do what frozen things do in
summer.

Kristoff: I'm gonna tell him.
Anna: Don't you dare!
Olaf: In summer!

Recomendación

9.1.2 Ayudar a los alumnos

Plazo: 03/12/19 - 15/12/19

Metas y acciones

9.1.2.1. Dar asesorías de ACARUS

9.1.2.1.1. curso express

9.1.2.1.2. curso intersemestral

9.1.2.1.3. materia optativa

9.1.2.2. Dar asesorías de ADOO

9.1.2.2.1. dar una hora de clase los sábados

9.1.2.2.2. fomentar la lectura de trabajos de tesis

9.1.2.2.3. regalar dinero

Evidencia

Título: Se dio un curso de 3 horas

Acción:
9.1.2.1.1. curso express

Descripción:
Se
le
dio
un

curso express a los alumnos

Fecha: 06/12/19

Evidencia

Título: Se fomentó la lectura de trabajos de tesis

Acción:
9.1.2.2.2. fomentar la lectura de trabajos de tesis

Descripción:
Se les proporcionó una tesis doctoral a los alumnos y se les pidió que la expusieran en la semana de matemáticas

Fecha: 31/12/19

Let It Go

The snow glows white
on the mountain tonight,
not a footprint to be seen.
A kingdom of isolation and it looks
like I'm the queen.
The wind is howling like this
swirling storm inside.
Couldn't keep it in, Heaven knows I tried.
Don't let them in, don't let them see.
Be the good girl you always have to be.
Conceal don't feel, don't let them know.
Well, now they know!

Let it go, let it go!
Can't hold it back anymore.
Let it go, let it go!
Turn away and slam the door.
I don't care what they're going to say.
Let the storm rage on.
The cold never bothered me anyway.

It's funny how some distance,
makes everything seem small.
And the fears that once controlled me,
can't get to me at all
It's time to see what I can do,
to test the limits and break through.

No right, no wrong, no rules for me.
I'm free!

Let it go, let it go.
I am one with the wind and sky.
Let it go, let it go.
You'll never see me cry.
Here I stand, and here I'll stay.
Let the storm rage on.

My power flurries through the air
into the ground.
My soul is spiraling in
frozen fractals all around
And one thought crystallizes like an icy blast
I'm never going back; the past is in the past!

Let it go, let it go.
And I'll rise like the break of dawn.
Let it go, let it go
That perfect girl is gone
Here I stand, in the light of day.

Let the storm rage on!
The cold never bothered me anyway.

Disney FROZEN

SONG LYRICS

Sing-along with the songs from Frozen!

In Summer

Kristoff: Really? I'm guessing you don't have much experience with heat.
Olaf: Nope! But sometimes I like to close my eyes, and imagine what it'll be like when summer does come.

Bees'll buzz, kids'll blow dandelion fuzz
And I'll be doing whatever snow does
in summer.
A drink in my hand,
my snow up against the burning sand
Prob'ly getting gorgeously tanned
in summer.

I'll finally see a summer breeze,
blow away a winter storm.
And find out what happens to solid water
when it gets warm!
And I can't wait to see,
what my buddies all think of
me.
Just imagine how much
cooler I'll be
in summer.

Dah dah, da doo,
uh bah bah bah bah bah boo

The hot and the cold are both so intense,
Put 'em together it just makes sense!

Rrr Raht da daht dah dah dah
dah dah dah dah dah doo

Winter's a good time
to stay in and cuddle,
But put me in summer and I'll be a —
happy snowman!

When life gets rough,
I like to hold on to my dream,
Of relaxing in the summer sun,
just lettin' off steam.

Oh the sky would be blue,
and you guys will be there too
When I finally do what frozen things do in
summer.

Kristoff: I'm gonna tell him.
Anna: Don't you dare!
Olaf: In summer!